Basic Research in Computer Science



# Modelling and Control of Discrete Event Dynamic Systems

František Čapkovič

BRICS Report Series RS-00-26

ISSN 0909-0878 October 2000

**See back inner page for a list of recent BRICS Report Series publications.**
**Copies may be obtained by contacting:**

**BRICS**
**Department of Computer Science**
**University of Aarhus**
**Ny Munkegade, building 540**
**DK–8000 Aarhus C**
**Denmark**
**Telephone: +45 8942 3360**
**Telefax: +45 8942 3255**
**Internet: BRICS@brics.dk**

**BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:**

`http://www.brics.dk`
`ftp://ftp.brics.dk`
**This document in subdirectory** `RS/00/26/`

# Modelling and Control of Discrete Event Dynamic Systems

František Čapkovič
*capkovic@brics.dk*

October, 2000

**Abstract**

Discrete event dynamic systems (DEDS) in general are investigated as to their analytical models most suitable for control purposes and as to the analytical methods of the control synthesis. The possibility of utilising both the selected kind of Petri nets and the oriented graphs on this way is pointed out. Because many times the control task specifications (like criteria, constraints, special demands, etc.) are given only verbally or in another form of non analytical terms, a suitable knowledge representation about the specifications is needed. Special kinds of Petri nets (logical, fuzzy) are suitable on this way too. Hence, the knowledge-based control synthesis of DEDS can also be examined. The developed graphical tools for model drawing and testing as well as for the automated knowledge-based control synthesis are described and illustratively presented.

Two approaches to modelling and control synthesis based on oriented graphs are developed. They are suitable when the system model is described by the special kind of Petri nets - state machines. At the control synthesis the first of them is straightforward while the second one combines both the straight-lined model dynamics development (starting from the given initial state towards the prescribed terminal one) and the backtracking model dynamics development.

# Contents

# 1 Introduction

In Control Theory there are many successful methods of modelling and control that are suitable for the continuous-time systems (CTS) or/and discrete-time systems (DTS). However, usually they are not usable for solving the problems connected with modelling and control of DEDS. Namely, DEDS are completely different (as to the principle of their dynamic behaviour) from the CTS and DTS. The development of their dynamic behaviour depends on the occurrence of discrete events. DEDS are asynchronous systems with many conflict situations and with high parallelism among subsystems activities. A typical course of a DEDS variable $x$ is given on the Fig. 1. It can be seen that there is not

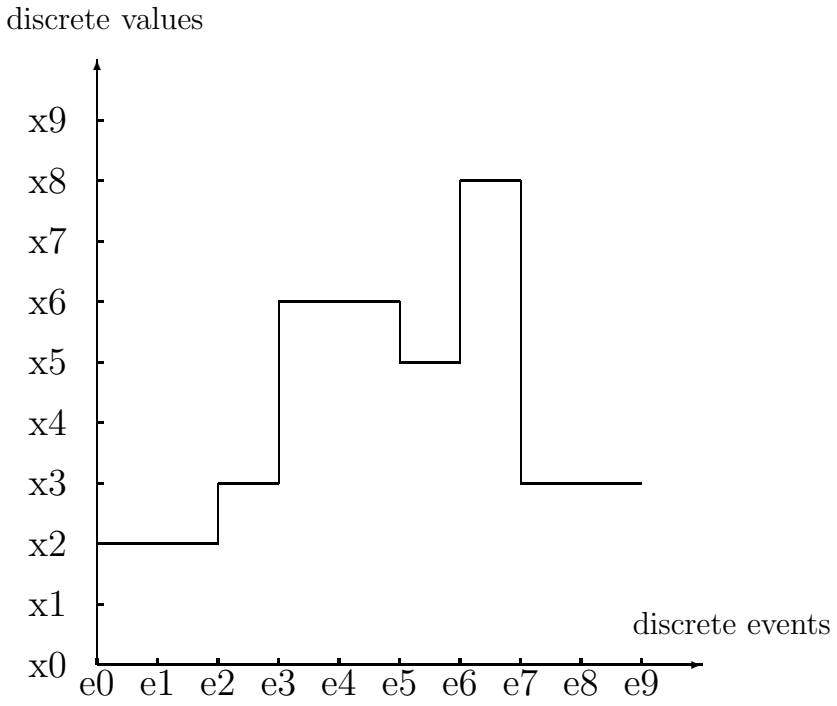


Figure 1: The course of a variable of DEDS

any explicit time on the horizontal axis but only the sequence of occurring events $\{e_0, e_2, e_3, e_5, e_6, e_7\}$ with the corresponding sequence $\{x_2, x_3, x_6, x_5, x_8, x_3\}$ of the variable values on the vertical axis representing responses on the discrete events occurrence. A different order of the events sequence leads to a different sequence of the values i.e. to

the different course of the variable $x$.

DEDS are used very frequently in human practice. Usually they are large-scale and/or complex systems. Typical representatives of DEDS are especially manufacturing systems, some kinds of transport systems and communication systems (including the communication processes inside computers). DEDS in general will be investigated here with the aim to find analytical models most suitable for control purposes and as to the analytical methods of the control synthesis. The possibility of utilising both the selected kinds of Petri nets (PN) and the oriented graphs (OG) on this way is pointed out. However, the manufacturing systems (MS), especially flexible manufacturing systems (FMS) will be emphasised a little more.

The system control in general is defined as the task when it is necessary to transfer the system from a given initial state into a prescribed terminal one at simultaneous satisfying control task specifications like criteria, constraints, external conditions, etc. imposed on the system activity. The specifications prescribes how the system should operate.

Control tasks for DEDS are usually multi-criterial and many times they are given only verbally or in another form of non-analytical terms. Consequently, DEDS need methods for their modelling and control that are different from those used for CTS and DTS. Especially, a suitable knowledge representation about the control task specifications is needed in order to quantify them. Therefore, the knowledge-based control synthesis of DEDS cannot be avoided. Special kinds of PN (logical, fuzzy) are used on this way.

However, in spite of the fact that the model is able to describe the system to be controlled, it does not yield any prescription how to control the system. Consequently, the control synthesis procedure should be found in order to deal with the DEDS control problems - i.e. to find the a way how to reach the prescribed terminal state from a given initial one and simultaneously satisfy the control task specifications.

The PN-based approach to modelling DEDS and the knowledge-based control synthesis are presented in the first part of this paper. The graphical tools developed for both the system modelling and the automated knowledge-based control synthesis are also described there.

The second part is devoted to the OG-based approaches suitable for the systems that can be modelled by the special kind of PN - the state machines. The methods make analytical solving the control synthesis problems possible. Such a process is automated or even fully automatic.

Two approaches are presented. The first procedure is straightforward, based on the functional adjacency matrix of OG. The second procedure combines both the straight-lined development (starting from the given initial state towards the prescribed terminal one) of the system model behaviour and the backtracking one (starting from the prescribed terminal state towards the given initial one). The possibility of creating the OG-based $k$-variant model of DEDS (where $k$ is the step of the model dynamics development) corresponding to the PN-based $k$-invariant one was pointed out already in the author's paper [2] and (in an extended form) recently in [11]. The main principle of the simple straightforward control system synthesis was also presented in the latter paper. Because such an approach to the control synthesis is not fully automatic, but only automated, the idea of the combination both the straight-lined approach and the backtracking one is unfolded. It even makes the automatic solving the control synthesis problem possible.

From the system theory point of view MS are a kind of DEDS because they meet all attributes mentioned above - they consist of many cooperating subsystems with many conflicts among some of them on one hand but also the parallelism among some of them on the other hand. Even, the possibility of the parallelism is welcome because of the maximal productivity criterion. The MS behaviour is influenced by occurring discrete events that start or stop activities of the subsystems (e.g. machine tools, robots, automatically guided vehicles, etc.). They are asynchronous systems. Usually, they are large-scale or/and complex. The presented methods are especially suitable for the MS subclass - for FMS. Because MS are very important in human practice, the demand of the successful and efficient control of them is very actual.

# 2 Petri net-based approach

## 2.1 Modelling DEDS

PN are utilised for modelling of many kind of systems [25]. PN-based models of DEDS are used very frequently. The approach based on an analogy with the ordinary PN (OPN) is used also here, in order to build the mathematical model of the DEDS to be controlled. It is the analogy between the DEDS subprocesses or activities and the OPN positions (the PN places will be named here as the positions) as well as the analogy between the DEDS discrete events and the OPN transitions.

### 2.1.1 The model structure

The OPN are understood here (as to their structure) to be the directed bipartite graphs

$$\langle P, T, F, G \rangle ; \quad P \cap T = \emptyset ; \quad F \cap G = \emptyset \tag{1}$$

where

$P = \{p_1, ..., p_n\}$ is a finite set of the OPN positions with $p_i$ , $i = 1, n$, being the elementary positions.

$T = \{t_1, ..., t_m\}$ is a finite set of the OPN transitions with $t_j$ , $j = 1, m$, being the elementary transitions.

$F \subseteq P \times T$ is a set of the oriented arcs entering the transitions. It can be expressed by means of the arcs incidence matrix $\mathbf{F} = \{f_{ij}\}$ , $f_{ij} \in \{0, M_{f_{ij}}\}$ , $i = 1, n$ ; $j = 1, m$. The element $f_{ij}$ represents the absence (when 0) or presence and multiplicity $M_{f_{ij}}$ (when $M_{f_{ij}} > 0$) of the arc oriented from the position $p_i$ to its output transition $t_j$.

$G \subseteq T \times P$ is a set of the oriented arcs emerging from the transitions. It can be expressed by means of the arcs incidence matrix $\mathbf{G} = \{g_{ij}\}$ , $g_{ij} \in \{0, M_{g_{ij}}\}$ , $i = 1, m$ ; $j = 1, n$.

The element $g_{ij}$ represents analogically (to the matrix F) the absence or presence and multiplicity of the arc oriented from the transition $t_i$ to its output position $p_j$.

$\emptyset$ is an empty set.

### 2.1.2 The model dynamics

However, OPN have not only their structure but also their dynamics - i.e. marking of their positions and its dynamic development. It can formally be expressed by another quadruplet

$$\langle X, U, \delta, \mathbf{x}_0 \rangle ; \quad X \cap U = \emptyset \tag{2}$$

where

$X = \{\mathbf{x}_0, \mathbf{x}_1..., \mathbf{x}_N\}$ is a finite set of the state vectors with $\mathbf{x}_k$ , $k = 0, N$, being the elementary state vectors. Here, $\mathbf{x}_k = (\sigma^k_{p_1}, ..., \sigma^k_{p_n})^T$ is the n-dimensional state vector (marking) of the OPN-based model in the step $k$ of the system dynamics development. The element $\sigma^k_{p_i} \in \{0, c_{p_i}\}$, $i = 1, n$ is the state of the elementary position $p_i$ in the step $k$ - i.e. the passivity (when 0) or activity (when $0 < \sigma^k_{p_i} \leq c_{p_i}$), where $c_{p_i}$ is the capacity of the position $p_i$. $(.)^T$ symbolises the matrix or vector transpose.

$U = \{\mathbf{u}_0, \mathbf{u}_1 ..., \mathbf{u}_N\}$ is a finite set of the elementary control vectors $\mathbf{u}_k$ , $k = 0, N$. Here, $\mathbf{u}_k = (\gamma^k_{t_1}, ..., \gamma^k_{t_m})^T$ is the m-dimensional control vector of the OPN-based model in the step $k$ and $\gamma^k_{t_j} \in \{0, 1\}$, $j = 1, m$ is the state of the elementary transition $t_j$ in the step $k$ - i.e. disabled (when 0) or enabled (when 1).

$\delta : X \times U \longrightarrow X$ is the transition function of the model dynamics development.

$\mathbf{x}_0$ is an initial state vector of the model.

The simplest form how to describe the transition function $\delta$ in analytical terms - see e.g. [3, 29] - is the following linear discrete system that will represent the DEDS model

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{B}.\mathbf{u}_k \quad , \quad k = 0, N \tag{3}$$

$$\mathbf{B} = \mathbf{G}^T - \mathbf{F} \tag{4}$$

$$\mathbf{F}.\mathbf{u}_k \leq \mathbf{x}_k \tag{5}$$

where

$k$ is the discrete step of the DEDS dynamics development

$\mathbf{x}_k = (\sigma^k_{p_1}, ..., \sigma^k_{p_n})^T$ is the $n$-dimensional state vector of the system in the step $k$. Its components express the states of the DEDS elementary subprocesses or operations - in case of MS e.g. waiting or movement of robots, tooling by a machine tool, number of the enter or exit parts on a pallet, etc. The capacity of the OPN position $p_i$ as to its marking represents e.g. the maximal number of technical parts that can be placed onto a pallet or a transport belt of MS.

$\mathbf{u}_k = (\gamma^k_{t_1}, ..., \gamma^k_{t_m})^T$ is the $m$-dimensional control vector of the system in the step $k$. Its components represent occurring of the DEDS elementary discrete events - e.g. starting or ending elementary operations, switching machines on/off and other activities.

$\mathbf{B}$, $\mathbf{F}$, $\mathbf{G}$ are, respectively, $(n \times m)$, $(n \times m)$ and $(m \times n)$- dimensional structural matrices of constant elements. The matrix $\mathbf{F}$ expresses the mutual causal relations among the states of the DEDS and the discrete events occurring during the DEDS operation, when the states are the causes and the events are the consequences. The matrix $\mathbf{G}$ expresses very analogically the causal relations among the discrete events (the causes) and the DEDS states (the consequences). Both of these matrices are (in the graph theory terminology) said to be the oriented arcs incidence matrices. The matrix $\mathbf{B}$ is given by (4).

## 2.2 The control synthesis

The control synthesis problem is that of finding the most suitable sequence (with respect to the control task specifications) of the control vectors $\mathbf{u}_k,\ k = 0, N$ that is able to transform the controlled system from the given initial state $\mathbf{x}_0$ to a prescribed terminal state $\mathbf{x}_t$. However as a rule, the DEDS control policy cannot be expressed analytically in a closed form like in CTS or DTS. Namely, the control task specifications (e.g. constraints, criteria, etc.) are usually expressed only verbally. Consequently, in order to quantify the specifications the proper knowledge representation (e.g. the rule-based one) is needed - see [3] - in the form of a domain oriented knowledge base. The knowledge base is utilised at the choice of the most suitable control vector $\mathbf{u}_k$ in any step $k$ when there are several possibilities at disposal (in order to avoid any ambiguity as to the further development of the DEDS dynamics).

### 2.2.1 The verbal flowchart of the procedure

In order to find the suitable control vector $\mathbf{u}_k$ able to transform the system from the existing state $\mathbf{x}_k$ into a following state $\mathbf{x}_{k+1}$ the simple procedure can be used. It can be concisely described as follows:

START

- $k = 0$ i.e. $\mathbf{x}_k = \mathbf{x}_0$; $\mathbf{x}_0$ is an initial state; $\mathbf{x}_t$ is a terminal state

LABEL:

- generation of the control base $\mathbf{w}_k$
- generation of the possible control vectors $\{\mathbf{u}_k\} \in \mathbf{w}_k$
- generation of the corresponding model responses $\{\mathbf{x}_{k+1}\}$
- consideration of the possibilities in the **knowledge base** (built on IF-THEN rules and expressing the control task specifications)
- choice of the most suitable control possibility
- *if* (the $\mathbf{x}_t$ or another stable state was found) *then (goto* END) *else ( begin* $k = k + 1$ ; *goto* LABEL; *end*)

END

This procedure is schematically illustrated on Fig. 2.

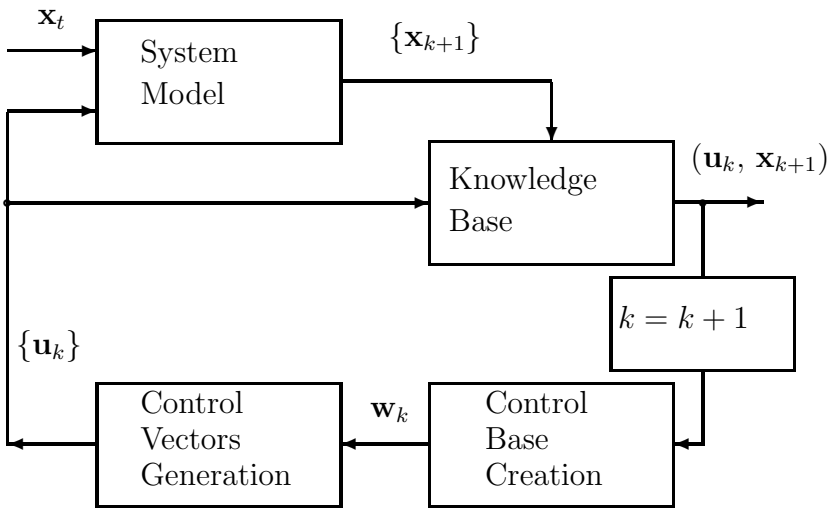


Figure 2: The principial procedure of the off-line control synthesis

### 2.2.2 The description of particulars

The procedure of the control base generation is the following

$$\mathbf{x}_k = ({x_1}^k, ..., {x_n}^k)^T \tag{6}$$

$$\mathbf{y}_k = ({y_1}^k, ..., {y_n}^k)^T \tag{7}$$

$${y_i}^k = \begin{cases} 1 & \text{if } {x_i}^k > 0 \\ 0 & \text{otherwise} \end{cases} \quad ; \quad i = 1, n \tag{8}$$

$$\overline{\mathbf{y}}_k = \underline{neg}\,\mathbf{y}_k = \mathbf{1}_n - \mathbf{y}_k \tag{9}$$

$$\mathbf{v}_k = \mathbf{F}^T \cdot \overline{\mathbf{y}}_k \tag{10}$$

$$\mathbf{v}_k = ({v_1}^k, ..., {v_m}^k)^T \tag{11}$$

$$\mathbf{z}_k = ({z_1}^k, ..., {z_m}^k)^T \tag{12}$$

$${z_j}^k = \begin{cases} 1 & \text{if } {v_j}^k > 0 \\ 0 & \text{otherwise} \end{cases} \quad ; \quad j = 1, m \tag{13}$$

$$\mathbf{w}_k = \underline{neg}\,\mathbf{z}_k = \mathbf{1}_m - \mathbf{z}_k \tag{14}$$

$$\mathbf{w}_k = ({w_1}^k, ..., {w_m}^k)^T \tag{15}$$

where

$\underline{neg}$ is the operator of logical negation.

$\overline{\mathbf{1}_n}$ is the $n$-dimensional constant vector with all of its elements equalled to the integer 1.

$\mathbf{y}_k$ is $n$-dimensional auxiliary vector with binary elements..

$\mathbf{v}_k$, $\mathbf{z}_k$ are, respectively, $m$-dimensional auxiliary vector and $m$-dimensional auxiliary vector with binary elements.

$\mathbf{w}_k$ is $m$-dimensional vector of the base for the control vector choice. The nonzero elements of the vector $\mathbf{y}_k$ point out the active positions. The nonzero elements of the vector $\overline{\mathbf{y}}_k$ point out the passive positions. The nonzero elements of the vector $\mathbf{z}_k$ point out the transitions having at least one passive position among their input positions. Hence, such transitions are disabled and from the control policy point of view they are out of the field of our interest. The nonzero elements of the vector $\mathbf{w}_k$ point out the OPN transitions that can theoretically (potentially) be enabled in the step $k$ - i.e. on the possible discrete events which could occur in the DEDS in the step $k$ and which could be utilised in order to transfer the system from the present state $\mathbf{x}_k$ into another state $\mathbf{x}_{k+1}$. The vector $\mathbf{w}_k$ represents the control base because it expresses the possible candidates for generating the control vectors $\{\mathbf{u}_k\}$ in the step $k$.

The above described procedure eliminates all disabled transitions. When only one of the $\mathbf{w}_k$ components is different from zero, it can be (when (5) is met, of course) used to be the control vector, i.e. $\mathbf{u}_k = \mathbf{w}_k$. The same is valid when $\mathbf{w}_k$ has more components but the condition (5) is fulfilled. The maximal parallelism is welcome in MS. When the condition (5) is not met and there are several components of the $\mathbf{w}_k$ different from zero, the most suitable control vector $\mathbf{u}_k$ has to be chosen on the base of additional information about the actual control task. The choice of the control vector can be made either by a human operator or automatically on the base of a corresponding domain oriented knowledge representation built in the form of the rules (e.g. IF-THEN ones) predefined by an expert in the corresponding domain. Such a knowledge base consists of a suitable expression of the constraints imposed upon the activities of the controlled system in question, control criteria, and further particulars concerning the control task.

In general, to obtain the elementary control vectors $\{\mathbf{u}_k\} \in \mathbf{w}_k$ the following procedure is performed

$$\begin{aligned} \mathbf{u}_k &= ({u_1}^k, \ ..., \ {u_m}^k)^T \\ \mathbf{u}_k &\subseteq \mathbf{w}_k \end{aligned} \tag{16}$$

$$u_j{}^k = \{ \begin{array}{ll} w_j{}^k & \text{if chosen} \\ 0 & \text{otherwise} \end{array} \; ; \; j = 1, m \tag{17}$$

Theoretically (i.e. from the combinatorics point of view) there exist

$$N_p{}^k = \sum_{i=1}^{N_t{}^k} \left( \begin{array}{c} N_t{}^k \\ i \end{array} \right) = 2^{N_t{}^k} - 1 \tag{18}$$

possibilities of the control vector choice in the step $k$. Here,

$$N_t{}^k = \sum_{j=1}^{m} \; w_j{}^k. \tag{19}$$

It means that there are the control vectors containing single nonzero elements of the base vector $\mathbf{w}_k$, the control vectors containing pairs of its nonzero elements, triples, quadruplets of its nonzero elements, etc., until the vector containing all of the nonzero elements of the base vector $\mathbf{w}_k$.

## 2.3 Knowledge representation

On the base of the above described control synthesis procedure, it is evident that a suitable form of the knowledge representation is needed in order to decide which control possibility should be actually chosen in any step $k$. To construct a suitable knowledge base (KB) the rule-based knowledge representation is usual in practice. The PN-based approach is used here in order to express the KB, even in analytical terms. It is supported by the logical PN (LPN) or/and fuzzy PN (FPN) - defined in [22], [23] and improved in [15].

### 2.3.1 The knowledge structure

Under the notion *knowledge* we mean some pieces of knowledge (some statements) mutually connected by causal interconnections in the form of IF-THEN rules. The statements are expressed by the LPN/FPN positions and the rules are expressed by the LPN/FPN transitions (taken together with their input and output positions). The mutual causality interconnections among the statements and rules are expressed by means of the analogy with the oriented arcs among the PN positions and transitions - see Fig. 3. More details about such a knowledge representation can be found in [4]-[10]. Consequently, the KB structure can be formally expressed as

$$\langle S, R, \Psi, \Gamma \rangle ; \quad S \cap R = \emptyset ; \quad \Psi \cap \Gamma = \emptyset \tag{20}$$

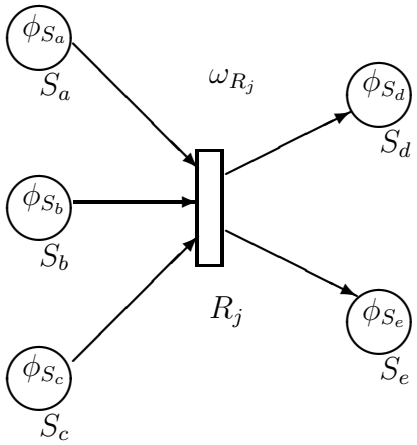


Figure 3: The rule $R_j$ with input and output statements

where

$S = \{S_1, ..., S_{n_1}\}$ is a finite set of the statements; $S_i$ , $i = 1, n_1$, are the pieces of knowledge (the elementary statements).

$R = \{R_1, ..., R_{m_1}\}$ is a finite set of the rules; $R_j$ , $j = 1, m_1$, are the rules either in the form of implications:

$R_j$ : $(S_a \, and \, S_b \, and \, ... \, and \, S_c) \Rightarrow (S_d \, and \, S_e)$

or in the form of IF-THEN structures:

$R_j$ : IF $(S_a \, and \, S_b \, and \, ... \, and \, S_c)$ THEN $(S_d \, and \, S_e)$,

where $S_a$, $S_b$, ..., $S_c$ are the input statements of the rule $R_j$, and the $S_d$, $S_e$ are the output statements of this rule.

$\Psi \subseteq S \times R$ is a set of the causal interconnections among the statements entering the rules (the causes) and the rules themselves. It can be expressed by means of the incidence matrix $\mathbf{\Psi} = \{\psi_{ij}\}$ , $i = 1, n_1$ ; $j = 1, m_1$. $\psi_{ij} \in \{0, 1\}$ in the analogy with the LPN and $\psi_{ij} \in < 0, 1 >$ in the analogy with the FPN. In other words the element $\psi_{ij}$ represents the absence (when 0), presence (when 1) or a fuzzy measure of existence (when its real value is between these boundary values) of the causal relation between the input statement $S_i$ and the rule $R_j$.

$\Gamma \subseteq R \times S$ is a set of the causal interconnections among the rules and the statements emerging from them (the consequences). It can be expressed by means of the incidence matrix $\mathbf{\Gamma} = \{\gamma_{ij}\}$ , $i = 1, m_1$ ; $j = 1, n_1$, where $\gamma_{ij} \in \{0, 1\}$ , in case of the LPN or $\gamma_{ij} \in < 0, 1 >$ in case

of the FPN. $\gamma_{ij}$ expresses the occurrence of the causal relation between the rule $R_i$ and its output statement $S_j$ (i.e. the absence, presence or a fuzzy measure of existence).

### 2.3.2 The knowledge dynamics

The KB dynamics development (i.e. the statements truth propagation) can be formally expressed as follows

$$\langle\, \Phi, \Omega, \delta_1, \mathbf{\Phi}_0 \,\rangle\,; \quad \Phi \cap \Omega = \emptyset \tag{21}$$

where

$\Phi = \{\mathbf{\Phi}_0,\ \mathbf{\Phi}_1 ...,\ \mathbf{\Phi}_{N_1}\}$ is a finite set of the KB elementary state vectors $\mathbf{\Phi}_K$ , $K = 0,\ N_1$. Here, $\mathbf{\Phi}_K = (\phi^K_{S_1},\ ...,\ \phi^K_{S_{n_1}})^T$ is the $n_1$-dimensional state vector of the statements truth propagation in the step $K$. $\phi^K_{S_i} \in \{0, 1\}$, $i = 1, n_1$ is the state of the elementary statement $S_i$ in the step $K$ - i.e. in the case of using the LPN-based model - true (when 1) or false (when 0). In the case of using the FPN-based model $\phi^K_{S_i} \in < 0, 1, >$, $i = 1, n_1$ and it expresses the fuzzy measure of the statement truth. $K$ is the step of the KB dynamics development.

$\Omega = \{\mathbf{\Omega}_0,\ \mathbf{\Omega}_1 ...,\ \mathbf{\Omega}_{N_1}\}$ is a finite set of the KB elementary control vectors $\mathbf{\Omega}_K$ , $K = 0, N_1$ expressing the state of the KB rules enabling. Here, $\mathbf{\Omega}_k = (\omega^K_{R_1},\ ...,\ \omega^K_{R_{m_1}})^T$ is the $m_1$-dimensional control vector of the KB in the step $k$. $\omega^K_{R_j} \in \{0, 1\}$, $j = 1, m_1$ is the state of the rule $R_j$ enabling in the step $K$ - i.e. disabled (when 0) or enabled (when 1). In fuzzy case $\omega^K_{R_j} \in < 0, 1 >$, $j = 1, m_1$ and it expresses the fuzzy measure of the rule enabling.

$\delta_1 : \Phi \times \Omega \longrightarrow \Phi$ is the transition function of the KB dynamics development.

$\mathbf{\Phi}_0$ is an initial state vector of the KB.

The simplest form how to describe the transition function $\delta_1$ in analytical terms is the following linear logical system that will represent the KB model.

$$\mathbf{\Phi}_{K+1} = \mathbf{\Phi}_K \,\underline{or}\, \mathbf{\Delta} \,\underline{and}\, \mathbf{\Omega}_K\,, \quad K = 0, N_1 \tag{22}$$

$$\mathbf{\Delta} = \mathbf{\Gamma}^T \,\underline{or}\, \mathbf{\Psi} \tag{23}$$

$$\mathbf{\Psi} \,\underline{and}\, \mathbf{\Omega}_K \leq \mathbf{\Phi}_K \tag{24}$$

where

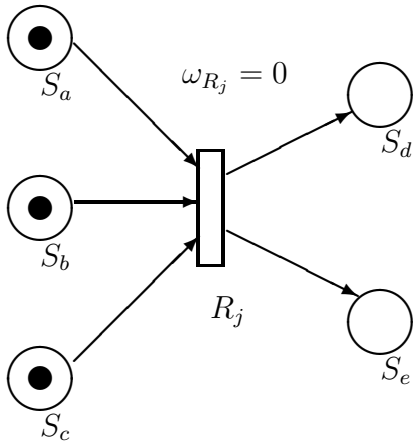


Figure 4: The enabled logical rule $R_j$

*and* is the operator of logical multiplying in general. For both the bivalued logic and the fuzzy one it can be defined (for scalar operands) to be the minimum of its operands. For example the result of its application on the scalar operands $a$, $b$ is a scalar $c$ which can be obtained as follows: $a \underline{and} b = c = min\{a, b\}$.

*or* is the operator of logical additioning in general. For both the bivalued logic and the fuzzy one it can be defined (for scalar operands) to be the maximum of its operands. For example the result of its application on the scalar operands $a$, $b$ is a scalar $c$ which can be obtained as follows : $a \underline{or} b = c = max\{a, b\}$.

## 2.4 The knowledge inference

The procedure of the knowledge inference is very analogical to that for obtaining the above introduced control base vector $\mathbf{w}_K$. It is the following

$$\overline{\mathbf{\Phi}}_K = \underline{neg}\,\mathbf{\Phi}_K = \mathbf{1}_{n_1} - \mathbf{\Phi}_K \tag{25}$$

$$\mathbf{v}_K = \mathbf{\Psi}^T \underline{and}\,\overline{\mathbf{\Phi}}_K \tag{26}$$

$$\begin{aligned} \mathbf{\Omega}_K &= \underline{neg}\,\mathbf{v}_K = \mathbf{1}_{m_1} - \mathbf{v}_K = \\ &= \underline{neg}(\mathbf{\Psi}^T \underline{and}\,(\underline{neg}\,\mathbf{\Phi}_K)) \end{aligned} \tag{27}$$

where

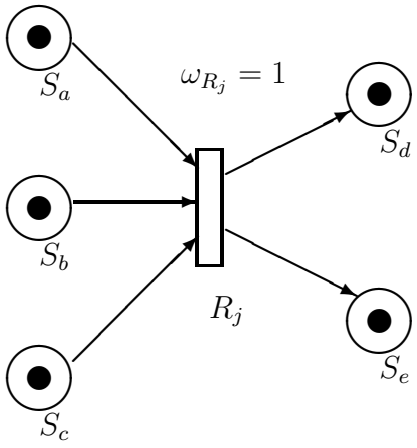


Figure 5: The logical rule $R_j$ after firing

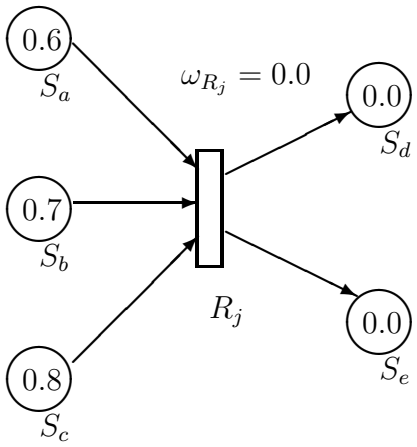


Figure 6: The enabled fuzzy rule $R_j$

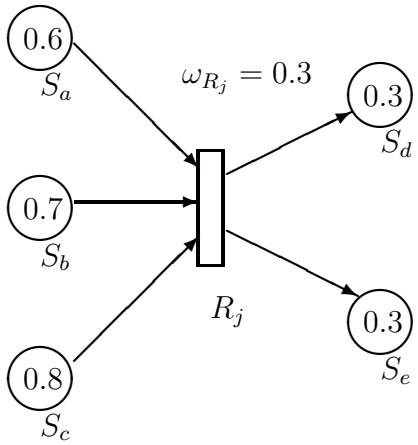


Figure 7: The fuzzy rule $R_j$ after firing

$\mathbf{v}_K$ is the $m_1$-dimensional auxiliary logical vector pointing out (by its nonzero elements) the rules that cannot be evaluated, because there is at least one false (of course in the LPN analogy) statement among its input statements.

$\mathbf{\Omega}_K$ is the $m_1$-dimensional "control" vector pointing out the rules that have all their input statements true and, consequently, they can be evaluated in the step $K$ of the KB dynamics development. This vector is a base of the inference, because it contains information about the rules that can contribute to obtaining the new knowledge - i.e. to transfer the KB from the state $\mathbf{\Phi}_K$ of the truth propagation into another state $\mathbf{\Phi}_{K+1}$. The rules are pointed out by the nonzero elements of the vector $\mathbf{\Omega}_K$.

$\underline{neg}$ is the operator of logical negation in general. For both the bivalued logic and the fuzzy one it can be defined (for scalar operands) to be the complement of its operand. For example : $\underline{neg}\, a = b = 1 - a$.

## 2.5 Graphical tools

To automatise the model creating and testing as well as the process of the knowledge-based control synthesis the graphical tools were developed.

### 2.5.1 The tool for the model drawing and testing

The graphical editor for drawing and testing the PN-based models is able to draw ordinary PN, to compute their invariants, to draw their reachability tree, to test their properties and to display the marking dynamics development (token player). In addition to this it is able to draw the logical and fuzzy PN-based models, the time and timed PN-based models (see e.g. [17, 18]) and to display their marking dynamics development. It was developed during works on Master Theses [1],[24],[16]. To illustrate some of its abilities, the following figures (see Fig. 8 - Fig. 13) are introduced.

### 2.5.2 The tool for the knowledge-based control synthesis

In order to automatise the control synthesis process as well as in order to bridge the OPN-based model with the LPN/FPN-based KB the program system (knowledge-based control synthesiser) was created in the Master Thesis [12]. It makes possible to express relations between the states of DEDS on one hand and the statements and rules of the KB on the other hand. It is user friendly and makes possible to simplify the work of the operator performing the DEDS control synthesis. Using the program system correctly, the control synthesis process can be fully automatised (even automatic).

Two files can be opened in the system - the OPN-based model (in the form of a file like 'model-name.pnt') created by means of the graphical editor of OPN mentioned above and the LPN/FPN-based KB (in the form of a file like 'kb-name.pnt') created by means of the same graphical editor by means of LPN/FPN. During the program system operation three windows are on the screen - see Fig. 14 or Fig. 15. The KB operates if the button KB is switched on. In the left window on the screen the OPN-based model is displayed (the graphical model or its verbal description can be alternatively seen). In the right window the LPN/FPN model of the KB or the I/O interface between OPN model and the KB can alternatively be seen. The I/O interface yields (at the beginning of its utilising) the empty skeleton corresponding to the number of the statements and rules of the KB. It can be fulfilled by the operator in order to define desirable relations between the OPN-based model and the LPN/FPN-based KB. For example in the case of the KB with only one rule with two input and one output statements the skeleton is the following:

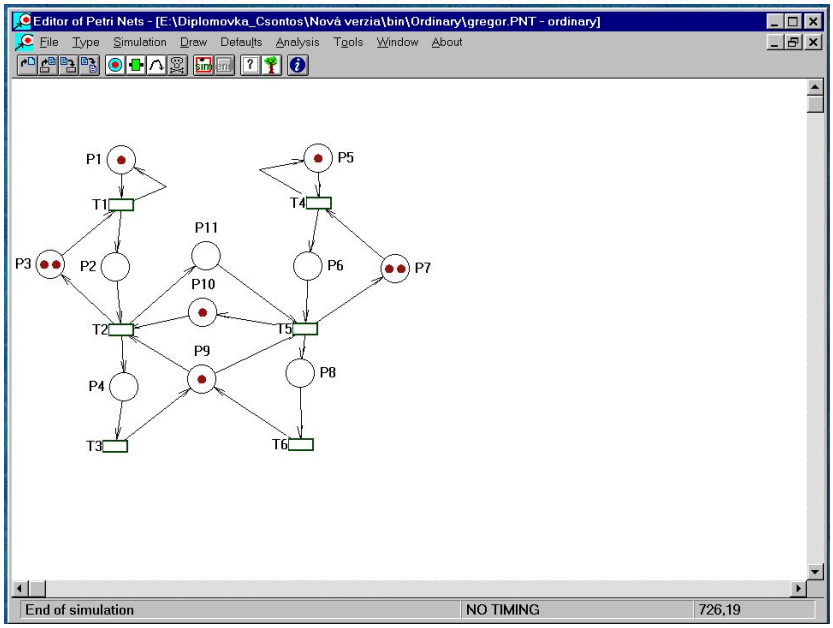


Figure 8: The example of the ordinary PN-based model.

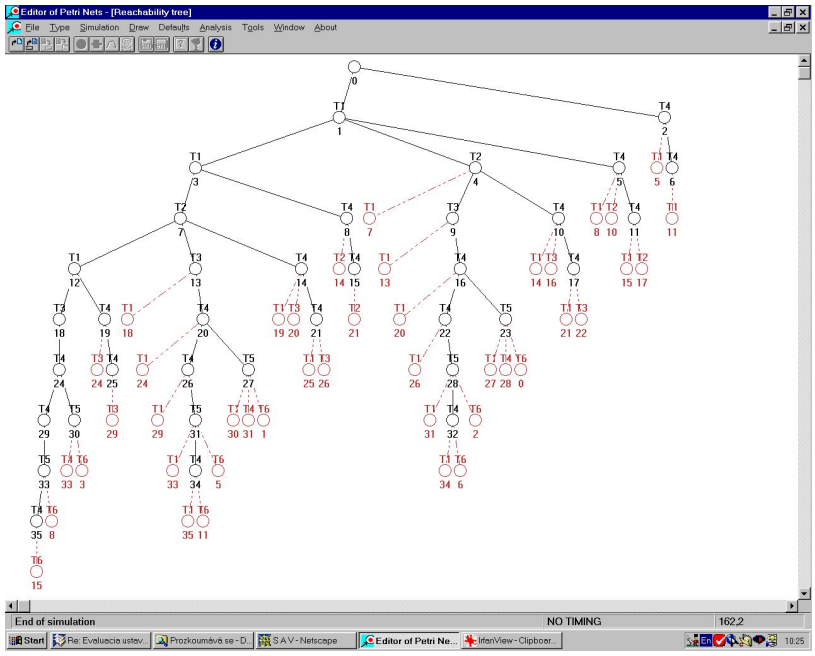


Figure 9: The reachability tree of this model.

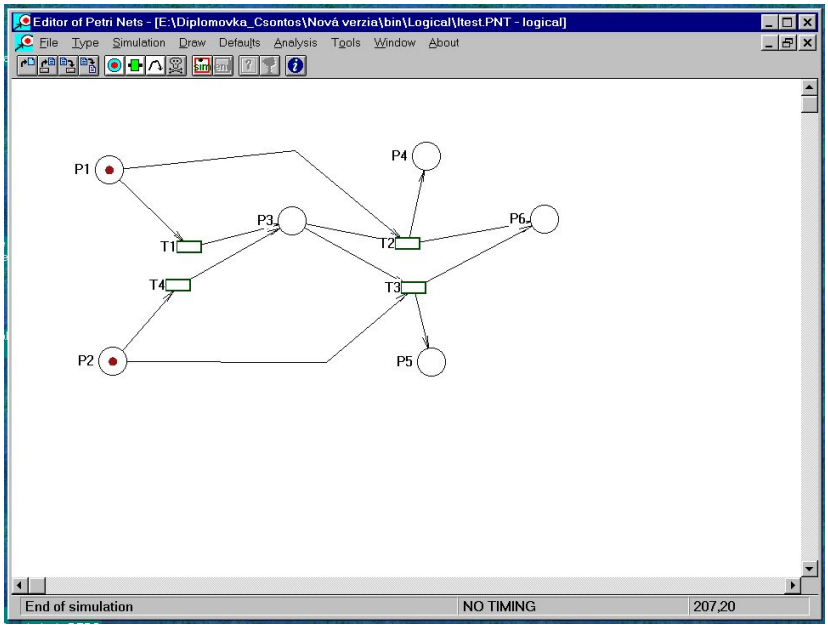


Figure 10: The example of the logical PN-based model.

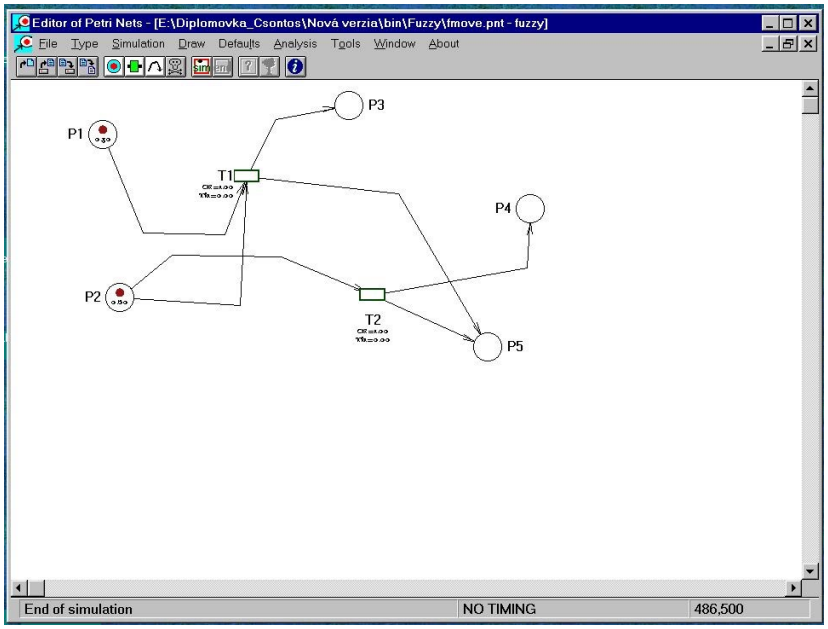


Figure 11: The example of the fuzzy PN-based model.

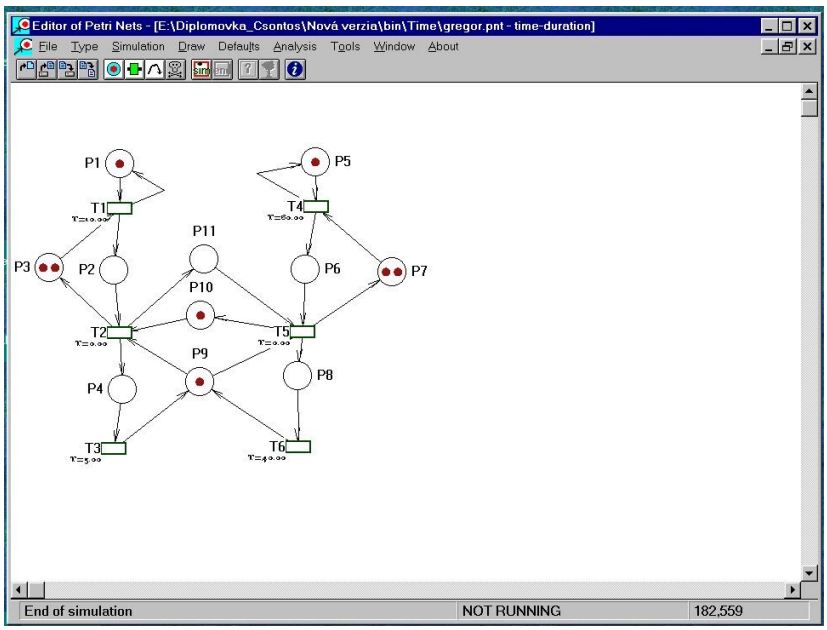


Figure 12: The example of the time PN-based model.

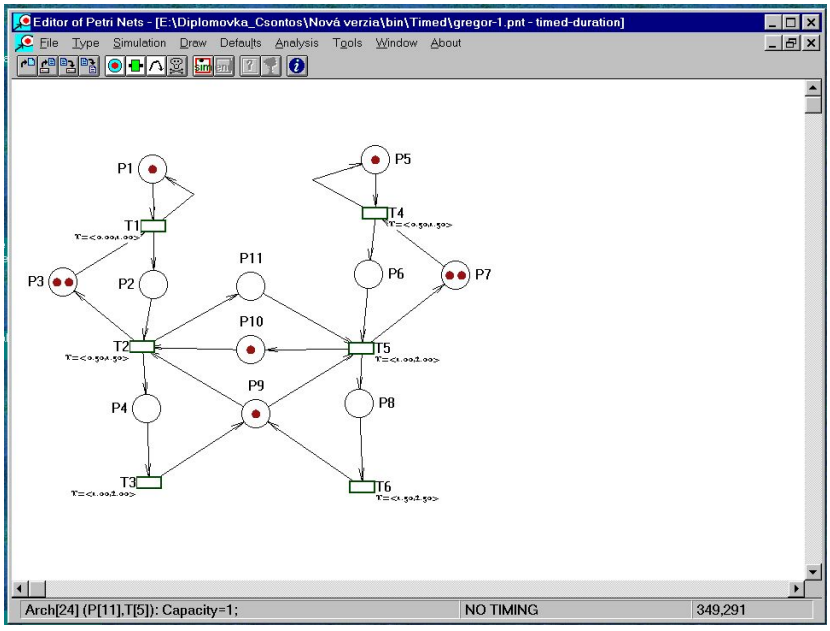


Figure 13: The example of the timed PN-based model.

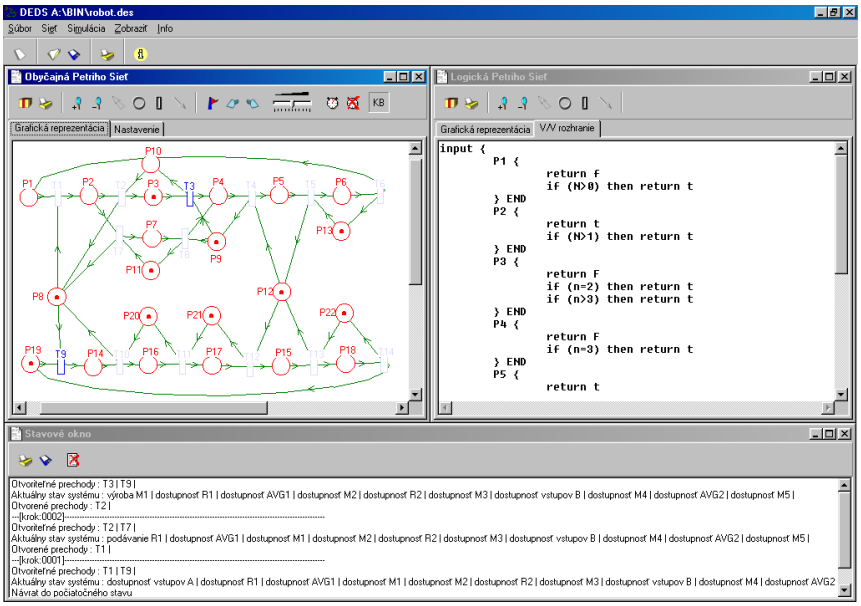


Figure 14: A view on the screen of control synthesiser - the graphical OPN model and the I/O interface.

```
input {
            P1 {
                return F
            } end
            P2 {
                return F
            } end
    } end
    output {
    // output place:
    // P3,
            return F
    } end
```

The third window (the state one) is placed in the down part of the screen and it contains the actual state of the system as well as information about the enabled transitions of the OPN model. After finishing the

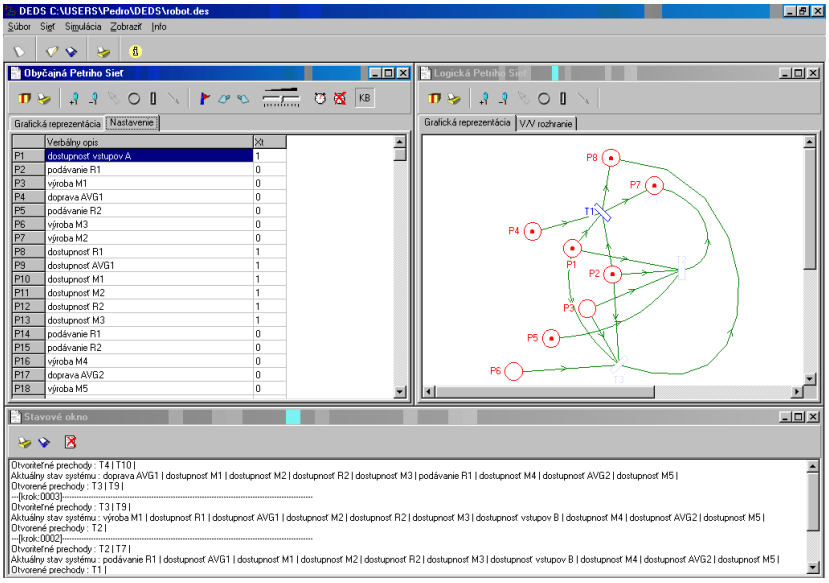


Figure 15: The verbal description of the OPN and KB.

control synthesis process the final sequence of the control interferences for the real DEDS can be obtained from this window. When KB is switched off, another small window appears in the center of the screen - see Fig. 16. It offers to the user the actual control possibilities and yields him the possibility to choose manually the most suitable one (from his subjective point of view). The same window appears also in the case when KB works but it is not able to choose the most suitable possibility. In such a case the decision must be made by the human operator. The detail description of the program system is given in the user handbook [13].

## 2.6 The illustrative example

In order to illustrate the above introduced approach consider the FMS given on the Fig. 17. It can be seen that it consists of two robots serving five machine tools, two automatic guided vehicles (AGVs), two entries (the inputs of raw materials A and B, respectively), and two exits (the outputs of the final A-parts and B-parts, respectively). The machines 1 and 2 produce the same intermediate A-parts and the machine 4 produces the intermediate B-parts. Machines 3 and 5 produce the final A-parts

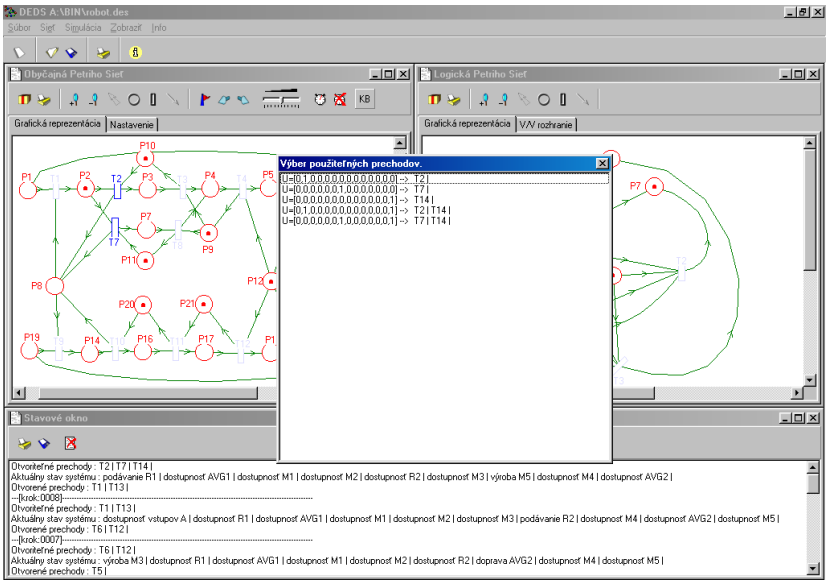


Figure 16: The window for the manual choice. KB is switched off.

and B-parts, respectively. Using above mentioned analogy the OPN-based model can be obtained - see Fig. 18.

It can be knitted e.g. by means of the method elaborated and presented in [14]. Meaning the OPN positions is the following:
P1 - availability of A-raw material, P2 - loading by Robot 1 (R1), P3 - machining by Machine 1 (M1), P4 - delivering via AGV1, P5 - loading by R2, P6 - machining by M3, P7 - machining by M2, P8 - availability of R1, P9 - availability of AGV1, P10 - availability of M1, P11 - availability of M2, P12 - availability of R2, P13 - availability of M3, P14 - loading by R1, P15 - loading by R2, P16 - machining by M4, P17 - delivering via AGV2, P18 - machining by M5, P19 - availability of B-raw material, P20 - availability of M4, P21 - availability of AGV2, P22 - availability of M5. The transitions T1 - T14 represent the starting or/and ending the corresponding operations. The nonzero elements of the structural matrices of the OPN-based model are in case of the $\mathbf{F}$ : $\{f_{11}, f_{22}, f_{27}, f_{33}, f_{44}, f_{55}, f_{66}, f_{78}, f_{81}, f_{89}, f_{93}, f_{98}, f_{10,2}, f_{11,7}, f_{12,4}, f_{12,12}, f_{13,5}, f_{14,10}, f_{15,13}, f_{16,11}, f_{17,12}, f_{18,14}, f_{19,9}, f_{20,10}, f_{21,11}, f_{22,13}\}$ and in case of the $\mathbf{G}$ : $\{g_{12}, g_{23}, g_{28}, g_{34}, g_{3,10}, g_{45}, g_{49}, g_{56}, g_{5,12}, g_{61}, g_{6,13}, g_{77}, g_{78}, g_{84}, g_{8,11}, g_{9,14}, g_{10,8}, g_{10,16}, g_{11,17}, g_{11,20}, g_{12,15}, g_{12,21}, g_{13,18}, g_{13,12}, g_{14,19}, g_{14,22}\}$. The full matrices are not presented with regard to the limited space. Starting

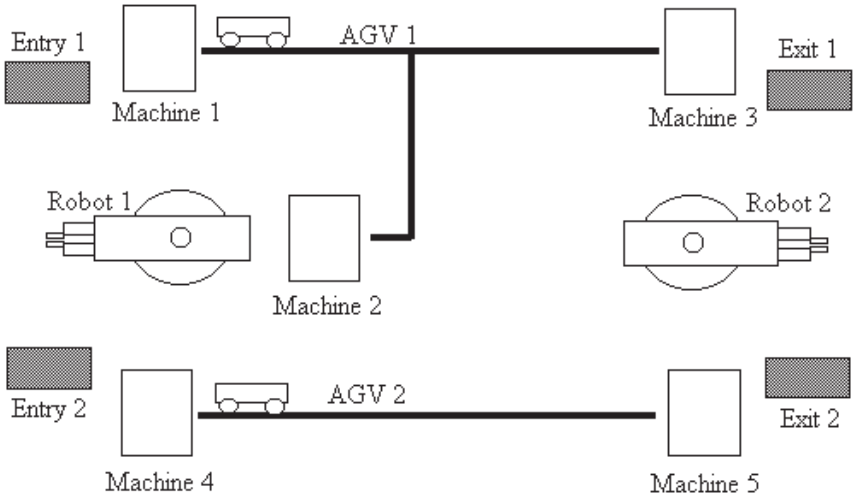


Figure 17: The flexible manufacturing system.

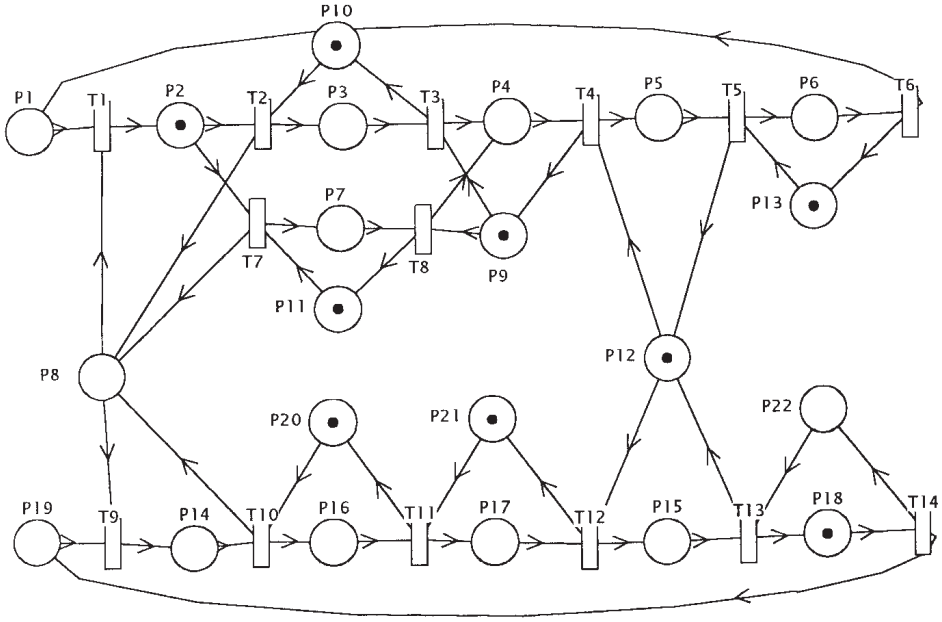


Figure 18: The OPN-based model of the FMS.

from the initial state vector of the process

$$\mathbf{x}_0 = (1\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1)^T$$

the following control base is obtained in the step $k = 0$

$$\mathbf{w}_0 = (1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0)^T$$

Hence, the following control possibilities can be automatically generated

$$\mathbf{u}_0^1 = (1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0)^T \tag{28}$$
$$\mathbf{u}_0^2 = (0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0)^T \tag{29}$$
$$\mathbf{u}_0^3 = (1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0)^T \tag{30}$$

Only $\mathbf{u}_0^3$ does not satisfy the existence condition (5). It means that remaining two possibilities are admissible, however, not simultaneously (R1 can take either A or B raw material). There is the conflict between them (i.e. between the enabled transitions T1 and T9). The model itself is not able to solve such a conflict because it has no information about it. To solve the conflict unambiguously external information (the intervention of the KB) is needed. The KB intervention should reflect both the actual state of the system and the external conditions (EC) expressing the control task specifications (e.g. the actual state of stores of the A and B raw materials or/and actual requirements on the amount of the production of the A and B final parts). The form of a simple rule can be (in case of $N_p$ control possibilities) e.g. the following: IF $((\mathbf{u}_k^1,\ \mathbf{x}_{k+1}^1)$ *and ... and* $(\mathbf{u}_k^i,\ \mathbf{x}_{k+1}^i)$ *and ... and* $(\mathbf{u}_k^{N_p},\ \mathbf{x}_{k+1}^{N_p})$ *and* EC) THEN ($\mathbf{u}_k^i$ corresponding to the EC).

When (28) is chosen in the step $k = 0$ then

$$\mathbf{x}_1 = (0\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1)^T$$
$$\mathbf{w}_1 = (0\,1\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0)^T$$

Hence, three control possibilities can be automatically generated. However, only two of them can be alternatively realized (R1 can serve either M1 or M2), namely

$$\mathbf{u}_1^1 = (0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0)^T \tag{31}$$
$$\mathbf{u}_1^2 = (0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0)^T \tag{32}$$

When the possibility (31) is chosen then

$$\mathbf{x}_2 = (0\,0\,1\,0\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1\,1)^T$$
$$\mathbf{w}_2 = (1\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0)^T$$

Consequently, seven control possibilities can be automatically generated. Five of them meet (5) and consequently, they can be separately realized. The following two ones must be eliminated

$$\mathbf{u}_2^5 = (1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0)^T$$
$$\mathbf{u}_2^7 = (1\,0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0)^T$$

When (29) is chosen in the step $k = 0$ then

$$\mathbf{x}_1 = (1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,0\,0\,0\,0\,0\,1\,1\,1)^T$$
$$\mathbf{w}_1 = (0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0)^T$$

Hence, only one control possibility is generated

$$\mathbf{u}_1 = (0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0)^T$$

It can be accepted because it satisfies (5). Consequently,

$$\mathbf{x}_2 = (1\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,0\,0\,1\,0\,0\,0\,0\,1\,1)^T$$
$$\mathbf{w}_2 = (1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0)^T$$

Hence, seven control possibilities can be automatically generated. Only five of them are admissible as to (5), however, their simultaneous using is excluded. The following two ones must be eliminated

$$\mathbf{u}_2^4 = (1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0)^T$$
$$\mathbf{u}_2^7 = (1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0)^T$$

It can be seen that the process is branching very extensively. To analyse all possibilities manually is practically impossible. In situations when there are several equivalent possibilities (from the theoretical point of view) how to choose the vector $\mathbf{u}_k$ in a step $k$ the domain oriented KB yields the most suitable one (from the practical point of view). Let us use the program system for the DEDS control synthesis. The proposed structure of the KB is given by the Fig. 19 and proposed corresponding interface is the following

```
input {
        P1 {
            return F
            if (N > 0) then return T
```

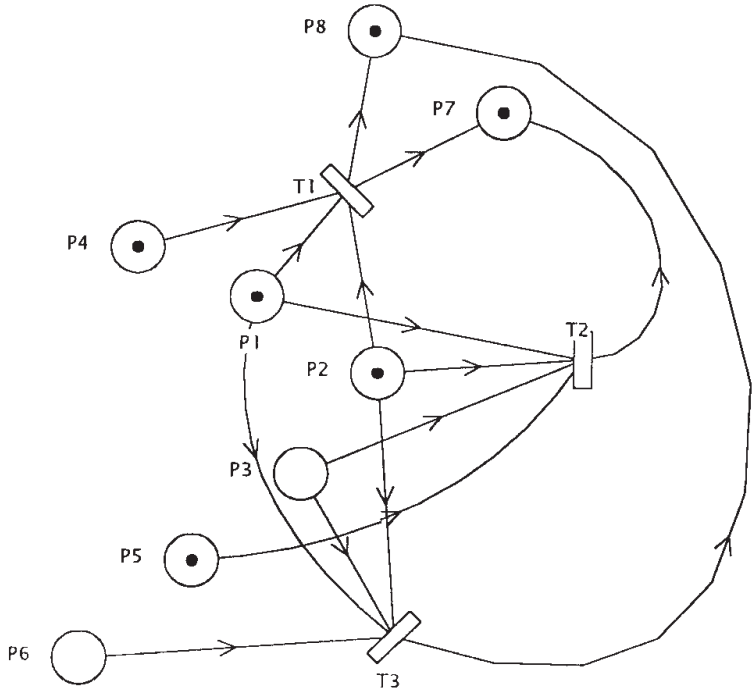


Figure 19: The KB of the FMS.

```
//N is the number of realizable control vectors in the
//actual step k of the OPN dynamics development
        } end
        P2 {
            return F
            if (N > 1) then return T
        } end
        P3 {
            return F
            if (N = 2) then return T
            if (N > 3) then return T
        } end
        P4 {
            return F
            if (N = 3) then return T
        } end
        P5 {
            return F
```

```
        } end
        P6 {
            return F
        } end
} end
output {
// output places:
// P7, P8
        return 0
        if (N < 4) then {
            if (p.7=1) & (p.8=0) then return 1
            if (p.7=0) & (p.8=1) then return 2
            if (p.7=1) & (p.8=1) then return 3
// p.j is the j-th component of the KB state vector
        }
        if (N > 3) then {
            if (p.7=1) & (p.8=0) then return 4
            if (p.7=0) & (p.8=1) then return 5
        }
        if (N = 1) then return 1
} end
```

The result of the program operation (when maximal parallelism is utilised) in the case when number of the final parts A has the priority with respect to the number of the final parts B is given in Tab. 1 as the sequence of control interferences into the real DEDS.

# 3 Graph-based approaches for the state machines

For analytical modelling and control synthesis of DEDS that can be described by the special kind of PN - the so called state machines (where any PN transition has only one input and only one output position)- the OG-based approaches are used.

## 3.1 The model based on the oriented graph

When the PN transitions are fixed on the corresponding oriented arcs among the PN positions - see Fig. 20 - we have a structure that can be

| step $k$ of dynamics | OPN fired transitions |
|---|---|
| 1 | $t_1$ |
| 2 | $t_2$ |
| 3 | $t_3$ & $t_9$ |
| 4 | $t_4$ & $t_{10}$ |
| 5 | $t_5$ & $t_{11}$ |
| 6 | $t_6$ & $t_{12}$ |
| 7 | $t_1$ & $t_{13}$ |
| 8 | $t_2$ & $t_{14}$ |
| 9 | $t_3$ & $t_9$ |
| . . . | . . . |

Table 1: The final results of the control synthesis.

understood to be the oriented graph. However, because the transition functions of elementary transitions were functions, the oriented arcs will be weighted by the functions too.

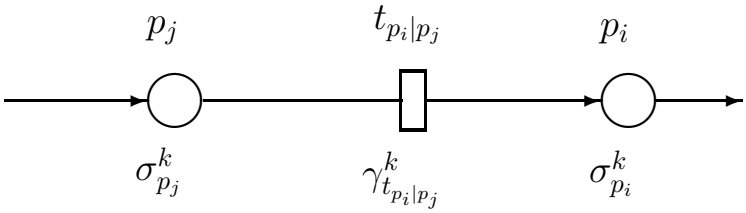


Figure 20: An example of the placement of a transition on the oriented arc between two positions $p_i$ and $p_j$

### 3.1.1 The model structure

The model structure can formally be described as

$$\langle P, \Delta \rangle \tag{33}$$

where

$P = \{p_1, ..., p_n\}$ is a finite set of the OG nodes with $p_i$ , $i = 1, n$, being the elementary nodes. As a matter of fact they are the PN positions.

$\Delta \subseteq P \times P$ is a set of the OG edges i.e. the oriented arcs among the nodes. Its elements are represented by the functions expressing the occurrence of the discrete events (represented above by the PN transitions). More precisely $\Delta = \Delta_k \subseteq (P \times T) \times (T \times P)$. It means that the PN transitions fixed on the oriented edges of the OG-based model represent the model parameters while in the PN-based model they represented the control vector. It is the principal difference between the PN-based model of DEDS and the OG-based one. The set $\Delta$ can be expressed in the form of the incidence matrix $\mathbf{\Delta}_k = \{\delta_{ij}^k\}$ , $\delta_{ij}^k \in \{0, 1\}$ , $i = 1, n$ , $j = 1, n$ , $k = 0,\ N$. Its element $\delta_{ij}^k$ represents the absence (when 0) or presence (when 1) of the edge oriented from the node $p_i$ to the node $p_j$ containing the PN transition. However, it is the transition function and its value depends also on the step $k$. Namely, the corresponding transition may be enabled in a step $k_1$ but disabled in another step $k_2$.

### 3.1.2 The model dynamics

The OG-based model dynamics can be formally expressed as follows

$$\langle X, \delta_1, \mathbf{x}_0 \rangle \tag{34}$$

where

$X = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N\}$ is a finite set of the state vectors of the graph nodes in different situations with $\mathbf{x}_k = (\sigma_{p_1}^k, ...,\ \sigma_{p_n}^k)^T$ , $k = 0, N$, being the $n$-dimensional state vector of the graph nodes in the step $k$; $\sigma_{p_i}^k \in \mathbf{x}_k$, $i = 1, n$ is the state of the elementary node $p_i$ in the step $k$; $k$ is the discrete step of the OG-based model dynamics development.

$\delta_1 : (X \times U) \times (U \times X) \longrightarrow X$ is the transition function of the graph dynamics. It contains implicitly the states of the transitions (the set $U$ is the same like in the PN-based model dynamics) situated on the OG edges.

$\mathbf{x}_0$ is the initial state vector of the model dynamics.

### 3.1.3 The OG-based model and its dynamics development

The $k$-variant OG-based linear discrete dynamic model of the DEDS can be written as follows

$$\{\mathbf{x}_{k+1}\} = \mathbf{\Delta}_k.\{\mathbf{x}_k\}\ ,\ k = 0, N-1 \tag{35}$$

where

$k$ is the discrete step of the DEDS dynamics development.

$\mathbf{x}_k = (\sigma^k_{p_1}, ..., \sigma^k_{p_n})^T$ ; $k = 0, N$ is in general the $n$- dimensional state vector of the DEDS in the step $k$; $\sigma^k_{p_i}$ , $i = 1, n$ is the state of the elementary subprocess $p_i$ in the step $k$. However, the model (35) generates aggregates of the state vectors because of its multiplicative character. Hence, it is better to write $\{\mathbf{x}_{k+1}\},\ k > 0,\ N-1$ as an aggregate of all of the states that are reachable from the previous aggregated state $\{\mathbf{x}_k\}$ in one step $k$; However, $\{\mathbf{x}_0\} = \mathbf{x}_0$, because the initial state is single.

$\mathbf{\Delta}_k = \{\delta^k_{ij}\}$ , $\delta^k_{ij} = \gamma^k_{t_{p_i|p_j}} \in \{0,\ 1\}$, $i = 1, n$ ; $j = 1, n$. This matrix expresses the causal relations between the subprocesses depending on the occurrence of the discrete events. The element $\delta^k_{ij} = \gamma^k_{t_{p_i|p_j}}$ expresses the actual value of the transition function of the PN transition fixed on the OG edge oriented from the node $p_j$ to the node $p_i$ in the step $k$.

Let us develop the system dynamics by means of the OG-based model in the straight-lined direction. Thus,

$$\begin{aligned}
\{\mathbf{x}_1\} &= \mathbf{\Delta}_0.\mathbf{x}_0 && (36)\\
\{\mathbf{x}_2\} &= \mathbf{\Delta}_1.\{\mathbf{x}_1\} = \mathbf{\Delta}_1.\mathbf{\Delta}_0.\mathbf{x}_0 && (37)\\
&\ \ \vdots \quad \vdots \quad \vdots && \\
\{\mathbf{x}_k\} &= \mathbf{\Delta}_{k-1}.\{\mathbf{x}_{k-1}\} = \mathbf{\Delta}_{k-1}\mathbf{\Delta}_{k-2}\ldots\mathbf{\Delta}_0.\mathbf{x}_0 && (38)\\
\{\mathbf{x}_k\} &= \mathbf{\Phi}_{k,0}.\mathbf{x}_0 && (39)\\
\{\mathbf{x}_N\} &= \mathbf{\Phi}_{N,0}.\mathbf{x}_0 && (40)\\
\mathbf{\Phi}_{k,j} &= \prod_{i=j}^{k-1}\mathbf{\Delta}_i \quad ; \quad j = 0, k-1 && (41)
\end{aligned}$$

where

$\mathbf{\Phi}_{k,j}$ is the transition matrix from the state $\mathbf{x}_j$ into the state $\mathbf{x}_k$. Multiplying of matrices inside the product symbol is made from the left because of respecting the causality principle in the model.

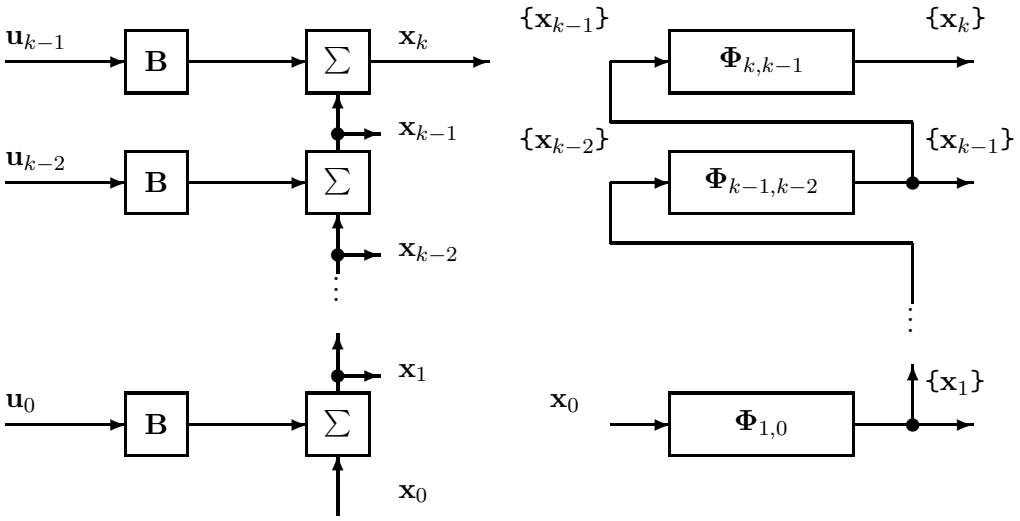


Figure 21: The illustration of comparing the PN-based model (on the left) and the OG-based one (on the right)

There is a symbolic interpretation as to the operators of multiplying and additioning inside the matrices. The element $\phi_{i,j}^{k,0} \in \mathbf{\Phi}_{k,0}$ has the form of either a product of $k$ elements (the transition functions expressing the sequence of elementary transitions that must be fired in order to reach the the final elementary state $\sigma_{p_i}^k$ from the initial elementary state $\sigma_{p_j}^0$) or a sum of several such products (when there are several ways how to reach the final state from the initial one). In the other words, any nonzero element $\delta_{ij}^k \in \mathbf{\Delta}_k$ yields information about reachability of the state $\sigma_{p_i}^{k+1}$ from the state $\sigma_{p_j}^k$. Thus, any element $\phi_{i,j}^{k_2,k_1} \in \mathbf{\Phi}_{k_2,k_1}$ yields information about the reachability of the state $\sigma_{p_i}^{k_2}$ from the state $\sigma_{p_j}^{k_1}$.

Both the PN-based model and the OG-based one are compared, as to their structure, on Fig. 21. It can be seen that the former model is strongly additive and it yields the actual states $\mathbf{x}_i$, $i = 0, k$ (however the control vectors $\mathbf{u}_i$, $i = 0, k-1$ has to be known) while the latter one is strongly multiplicative and it yields only aggregates of the state vectors, however the actual values of the parameters need not be known. Such an OG-based approach to the control synthesis yields the analytical solution in the closed form.

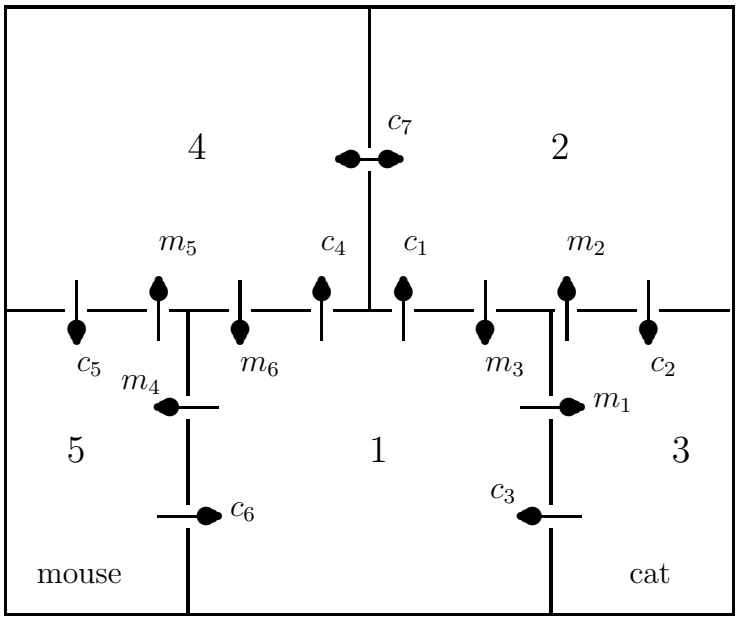


Figure 22: The maze structure.

### 3.1.4 The illustrative example

Consider the maze problem introduced by Ramadge and Wonham in [31]. Two participants - in [31] a cat and a mouse - can be as well e.g. two mobile robots or two AGVs of the FMS, two cars in a road network, two trains in a railway network, etc. They are placed in the maze (however, it can also be e.g. a complicated crossroad, complicated railways points, crossing AGVs lines in FMS etc.) given on Fig. 22 consisting of five rooms denoted by numbers 1, 2,..., 5 connecting by the doorways exclusively for the cat denoted by $c_i, i = 1, 7$ and the doorways exclusively for the mouse denoted by $m_j, j = 1, 6$. The cat is initially in the room 3 and the mouse in the room 5. Each doorway can be traversed only in the direction indicated. Each door (with the exception of the door $c_7$) can be opened or closed by means of control actions. The door $c_7$ is uncontrollable (or better, it is continuously open in both directions). The controller to be synthesised observes only discrete events generated by sensors in the doors. They indicate that a participant is just running through. The control problem is to find a feedback controller (e.g. an automatic points-man or switchman in railways, a system of crossroad lights, etc.) such that the following control task specifications - three criteria or/and

constraints will be satisfied:

1. The participants never occupy the same room simultaneously
2. It is always possible for both of them to return to their initial positions (the first one to the room 3 and the second one to the room 5)
3. The controller should enable the participants to behave as freely as possible with respect to the constraints imposed.

It can be seen that the criteria and constraints are completely given in the verbal form.

At the construction of the PN-based model of the system the rooms 1 - 5 of the maze will be represented by the PN positions $p_1$ - $p_5$ and the doorways will be represented by the PN transitions. The permanently open door $c_7$ is replaced by means of two PN transitions $t_7$ and $t_8$ symbolically denoted as $c_7^k$ and $c_8^k$. The PN-based representation of the maze is given on Figure 23. The initial state vectors of the cat and the mouse are

$$ {}^c\mathbf{x}_0 = (0\,0\,1\,0\,0)\ ,\ {}^m\mathbf{x}_0 = (0\,0\,0\,0\,1)^T \tag{42} $$

The structure of the cat and mouse control vectors is

$$ \begin{aligned} {}^c\mathbf{u}_k &= (c_1^k,\ c_2^k,\ c_3^k,\ c_4^k,\ c_5^k,\ c_6^k,\ c_7^k,\ c_8^k)^T;\ c_i^k \in \{0,1\},\ \ i = 1,8 \\ {}^m\mathbf{u}_k &= (m_1^k,\ m_2^k,\ m_3^k,\ m_4^k,\ m_5^k,\ m_6^k)^T;\ m_i^k \in \{0,1\},\ \ i = 1,6 \end{aligned} $$

The parameters of the cat model are

$$ n = 5 \qquad m_c = 8 $$

$$ {}^c\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}; {}^c\mathbf{G} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} $$

and the parameters of the mouse model are

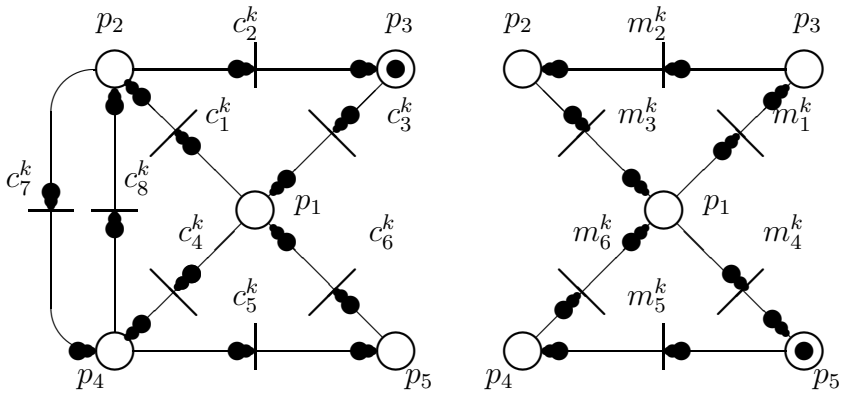


Figure 23: The PN-based representation of the maze. a) possible behaviour of the cat; b) possible behaviour of the mouse

$$n = 5 \qquad m_m = 6$$

$$^m\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}; {}^m\mathbf{G}_m^T = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

At the construction of the OG-based model (see Fig. 24) the matrices $^c\mathbf{\Delta}_k$ and $^m\mathbf{\Delta}_k$ of the system parameters are the following

$$^c\mathbf{\Delta}_k = \begin{pmatrix} 0 & 0 & c_3^k & 0 & c_6^k \\ c_1^k & 0 & 0 & c_8^k & 0 \\ 0 & c_2^k & 0 & 0 & 0 \\ c_4^k & c_7^k & 0 & 0 & 0 \\ 0 & 0 & 0 & c_5^k & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & {}^c\delta_{13}^k & 0 & {}^c\delta_{15}^k \\ {}^c\delta_{21}^k & 0 & 0 & {}^c\delta_{24}^k & 0 \\ 0 & {}^c\delta_{32}^k & 0 & 0 & 0 \\ {}^c\delta_{41}^k & {}^c\delta_{42}^k & 0 & 0 & 0 \\ 0 & 0 & 0 & {}^c\delta_{54}^k & 0 \end{pmatrix}$$

$$^m\mathbf{\Delta}_k = \begin{pmatrix} 0 & m_3^k & 0 & m_6^k & 0 \\ 0 & 0 & m_2^k & 0 & 0 \\ m_1^k & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & m_5^k \\ m_4^k & 0 & 0 & 0 & 0 \end{pmatrix} =$$

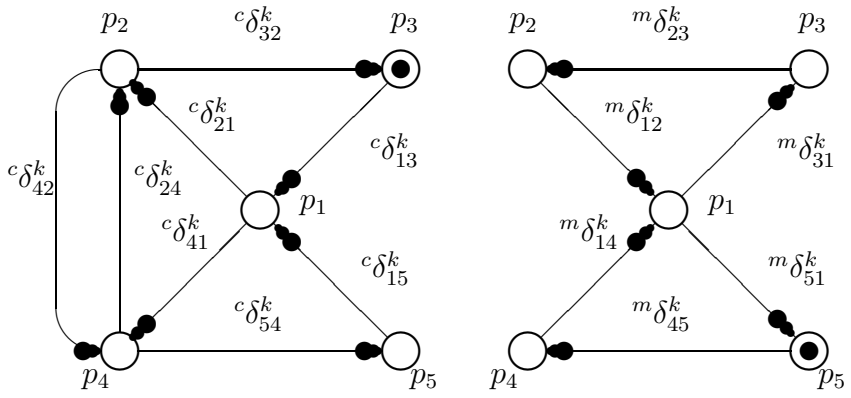


Figure 24: The OG-based model of the maze. a) possible behaviour of the cat; b) possible behaviour of the mouse

$$= \begin{pmatrix} 0 & {}^m\delta^k_{12} & 0 & {}^m\delta^k_{14} & 0 \\ 0 & 0 & {}^m\delta^k_{23} & 0 & 0 \\ {}^m\delta^k_{31} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & {}^m\delta^k_{45} \\ {}^m\delta^k_{51} & 0 & 0 & 0 & 0 \end{pmatrix}$$

The transitions matrices for the cat and mouse are the following

$${}^c\mathbf{\Phi}_{k+2,k} = {}^c\mathbf{\Delta}_{k+1}.{}^c\mathbf{\Delta}_k =$$

$$= \begin{pmatrix} 0 & c_3^{k+1}.c_2^k & 0 & c_6^{k+1}.c_5^k & 0 \\ c_8^{k+1}.c_4^k & c_8^{k+1}.c_7^k & c_1^{k+1}.c_3^k & 0 & c_1^{k+1}.c_6^k \\ c_2^{k+1}.c_1^k & 0 & \boxed{0} & c_2^{k+1}.c_8^k & 0 \\ c_7^{k+1}.c_1^k & 0 & c_4^{k+1}.c_3^k & c_7^{k+1}.c_8^k & c_4^{k+1}.c_6^k \\ c_5^{k+1}.c_4^k & c_5^{k+1}.c_7^k & 0 & 0 & \boxed{0} \end{pmatrix}$$

$${}^c\mathbf{\Phi}_{k+3,k} = {}^c\mathbf{\Delta}_{k+2}.{}^c\mathbf{\Delta}_{k+1}.{}^c\mathbf{\Delta}_k =$$

$$= \begin{pmatrix} c_3^{k+2}.c_2^{k+1}.c_1^k + c_6^{k+2}.c_5^{k+1}.c_4^k & c_6^{k+2}.c_5^{k+1}.c_7^k & \vdots \\ c_8^{k+2}.c_7^{k+1}.c_1^k & c_1^{k+2}.c_3^{k+1}.c_2^k & \vdots \\ c_2^{k+2}.c_8^{k+1}.c_4^k & c_2^{k+2}.c_8^{k+1}.c_7^k & \vdots \\ c_7^{k+2}.c_8^{k+1}.c_4^k & c_4^{k+2}.c_3^{k+1}.c_2^k + c_7^{k+2}.c_8^{k+1}.c_7^k & \vdots \\ c_5^{k+2}.c_7^{k+1}.c_1^k & 0 & \vdots \end{pmatrix}$$

$$\left.\begin{array}{cccc}
\vdots & 0 & c_3^{k+2}.c_2^{k+1}.c_8^k & 0 \\
\vdots & c_8^{k+2}.c_4^{k+1}.c_3^k & c_1^{k+2}.c_6^{k+1}.c_5^k + c_8^{k+2}.c_2^{k+1}.c_8^k & c_8^{k+2}.c_4^{k+1}.c_6^k \\
\vdots & \boxed{c_2^{k+2}.c_1^{k+1}.c_3^k} & 0 & c_2^{k+2}.c_1^{k+1}.c_6^k \\
\vdots & c_7^{k+2}.c_1^{k+1}.c_3^k & c_4^{k+2}.c_6^{k+1}.c_5^k & c_7^{k+2}.c_1^{k+1}.c_6^k \\
\vdots & c_5^{k+2}.c_4^{k+1}.c_3^k & c_5^{k+2}.c_7^{k+1}.c_8^k & \boxed{c_5^{k+2}.c_4^{k+1}.c_6^k}
\end{array}\right)$$

$${}^m\mathbf{\Phi}_{k+2,k} = {}^m\mathbf{\Delta}_{k+1}.{}^m\mathbf{\Delta}_k =$$

$$= \left(\begin{array}{ccccc}
0 & 0 & m_3^{k+1}.m_2^k & & m_6^{k+1}.m_5^k \\
m_2^{k+1}.m_1^k & 0 & 0 & 0 & 0 \\
0 & m_1^{k+1}.m_3^k & \boxed{0} & m_1^{k+1}.m_6^k & 0 \\
m_5^{k+1}.m_4^k & 0 & 0 & 0 & 0 \\
0 & m_4^{k+1}.m_3^k & 0 & m_4^{k+1}.m_6^k & \boxed{0}
\end{array}\right)$$

$${}^m\mathbf{\Phi}_{k+3,k} = {}^m\mathbf{\Delta}_{k+2}.{}^m\mathbf{\Delta}_{k+1}.{}^m\mathbf{\Delta}_k =$$

$$= \left(\begin{array}{cc:c}
m_3^{k+2}.m_2^{k+1}.m_1^k + m_6^{k+2}.m_5^{k+1}.m_4^k & 0 & \vdots \\
0 & m_2^{k+2}.m_1^{k+1}.m_3^k & \vdots \\
0 & 0 & \vdots \\
0 & m_5^{k+2}.m_4^{k+1}m_3^k & \vdots \\
0 & 0 & \vdots
\end{array}\right)$$

$$\left.\begin{array}{ccc}
0 & 0 & 0 \\
0 & m_2^{k+2}.m_1^{k+1}.m_6^k & 0 \\
\boxed{m_1^{k+2}.m_3^{k+1}.m_2^k} & 0 & m_1^{k+2}.m_6^{k+1}.m_5^k \\
0 & m_5^{k+2}.m_4^{k+1}.m_6^k & 0 \\
m_4^{k+2}.m_3^{k+1}.m_2^k & 0 & \boxed{m_4^{k+2}.m_6^{k+1}.m_5^k}
\end{array}\right)$$

The corresponding states reachability trees are given on Fig. 25 and Fig. 26. It can be seen that in order to fulfil the prescribed control task specifications introduced above, the comparison of the transition matrices of both animals in any step of their dynamics development is sufficient. Because the animals start from the defined rooms given by their initial states, it is sufficient to compare the columns 3 and 5. Consequently,

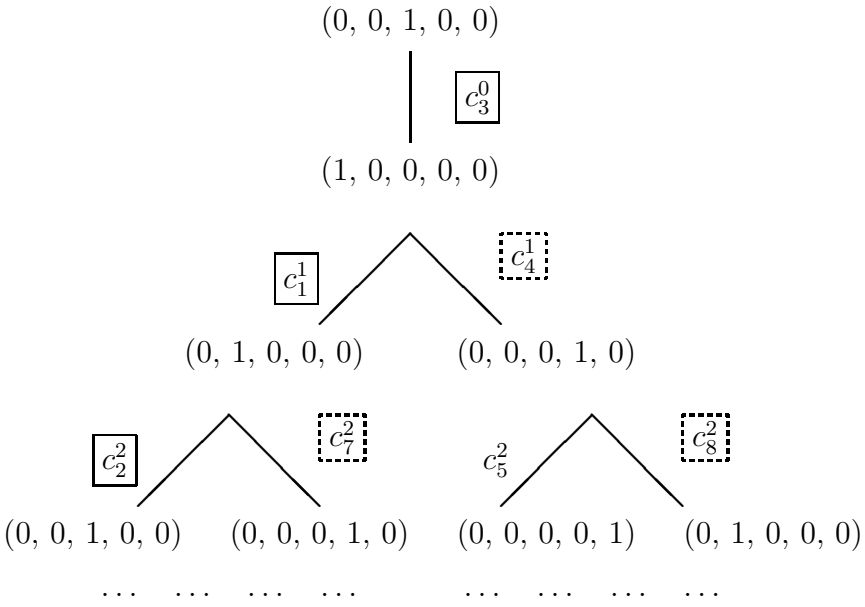


Figure 25: The fragment of the reachability tree of the cat

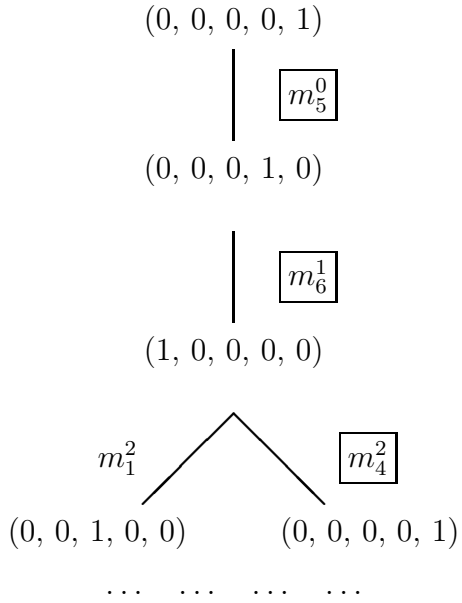


Figure 26: The reachability tree of the mouse

1. the corresponding (as to indices) elements of the transition matrices in these columns have to be mutually disjunct in any step of the dynamics development in order to avoid encounter of the animals on the corresponding trajectories.

2. if they are not disjunct one of them must be removed. It depends on the control task specifications which one will be removed. Let us go to focus attention on the elements with indices [3,3] and [5,5] of the matrices $\mathbf{\Phi}_{k+3,0}$. Namely, they express the trajectories making the return of the animals to their initial states possible. In case of the elements with indices [3,3] the element of the matrix ${}^{c}\mathbf{\Phi}_{k+3,0}$ should be chosen. It represents the trajectory of the cat making their come back possible. In case of the elements with indices [5,5] the element of the matrix ${}^{m}\mathbf{\Phi}_{k+3,0}$ should be chosen. It represents the trajectory of the mouse making their come back possible.

3. in the matrix ${}^{c}\mathbf{\Phi}_{k+3,k}$ two elements in the column 3 (with the indices [2,3] and [4,3]) stay unremoved, because of the permanently open door. It can be seen that also the elements of the column 5 of this matrix (with indices [2,5] and [4,5]) stay unremoved. This facts correspond with the prescribed condition 3 in the control task specifications.

On the base of these particulars it is clear that the KB construction need not be very complicated. In any case it is easier than that required by the OPN-based model at the knowledge-based approach to the control synthesis, because it is sufficient here to check only the actual elements of the transition matrices.

## 3.2 The combined approach to the control synthesis

The main disadvantage of the previous approach is the consumption of much memory because of the functional matrices - the functional adjacency matrix and its powers. It should be more suitable to work with the classical numerical adjacency matrix. Below such an approach is proposed. The adjacency matrix helps to generate automatically the state reachability tree in both the straight-lined system development (from an initial state towards the prescribed terminal one) and that of the backtracking system development (from the terminal state towards the initial one). To perform the DEDS control synthesis the combination both of

these kinds of the model behaviour development is used. The coincidence (a suitable intersection) both of the state reachability trees yields the possible trajectories of the system development. In such a way all solutions how to reach the prescribed terminal state from the given initial one are automatically found. To choose the most suitable solution rule-based knowledge about the control task specifications can also be utilised. Sometimes it may be fuzzy.

### 3.2.1 The straight lined dynamics development

When the input vector $\mathbf{x}_k$ represents a state of the system and we do not know the actual state of the transition functions (i.e. the actual state of the functional elements $\delta_{ij}^k$ of the $k$-variant matrix $\mathbf{\Delta}_k$), we can use the transpose of the classical (numerical) OG adjacency matrix - i.e. the matrix $\mathbf{\Delta} = \{\delta_{ij}\}$ having the same structure like the matrix $\mathbf{\Delta}_k = \{\delta_{ij}^{(k)}\}$, however its elements are not functional but they are defined as follows

$$\mathbf{\Delta} = \{\delta_{ij}\}; \quad \delta_{ij} = \{ \begin{array}{ll} 1 & \text{if } \delta_{ij}^k \neq 0 \\ 0 & \text{otherwise} \end{array} \quad ; \quad i = 1, n, \; j = 1, n \tag{43}$$

It means that $\mathbf{\Delta}$ is the constant matrix with all elements corresponding with the functional elements of the matrix $\mathbf{\Delta}_k$ equal to 1. Thus, in the below system development we can understand that $\mathbf{\Delta}_k = \mathbf{\Delta}$, $k = 0, N-1$. Let us start to derive the approach from the following form of the model description

$$\{\mathbf{x}_{k+1}\} = \mathbf{\Delta}.\{\mathbf{x}_k\} \; , \; k = 0, N-1 \tag{44}$$

where $\{\mathbf{x}_{k+1}\}$ is an aggregate of all of the states that are reachable from the previous states $\{\mathbf{x}_k\}$ in one step $k$. There is only one exception $\{\mathbf{x}_0\} = \mathbf{x}_0$, because the given initial state is only single. There is only one difference here in comparison with the model (35). The matrix $\mathbf{\Delta}$ is the constant matrix (the transpose of the adjacency matrix). Let us develop the system dynamics in the straight-lined orientation. Hence,

$$\begin{aligned}
\{\mathbf{x}_1\} &= \mathbf{\Delta}.\mathbf{x}_0 && (45)\\
\{\mathbf{x}_2\} &= \mathbf{\Delta}.\{\mathbf{x}_1\} = \mathbf{\Delta}.(\mathbf{\Delta}.\mathbf{x}_0) = \mathbf{\Delta}^2.\mathbf{x}_0 && (46)\\
&\vdots \quad \vdots \quad \vdots \\
\{\mathbf{x}_k\} &= \mathbf{\Delta}.\{\mathbf{x}_{k-1}\} = \mathbf{\Delta}.(\mathbf{\Delta}^{k-1}.\mathbf{x}_0) = \mathbf{\Delta}^k.\mathbf{x}_0 \\
\{\mathbf{x}_k\} &= \mathbf{\Phi}_{k,0}.\mathbf{x}_0 && (47)\\
\mathbf{\Phi}_{k,j} &= \prod_{i=j}^{k-1} \mathbf{\Delta} && (48)
\end{aligned}$$

The multiplying is made from the left because of the causality principle.

### 3.2.2 The backtracking dynamics development

The procedure very analogical to that starting from the initial state $\mathbf{x}_0$ can start from the terminal state $\mathbf{x}_t$. Let us denote the terminal state as $\mathbf{x}_N$. Using the functional matrix $\mathbf{\Delta}_k$, in general the following relation can be written

$$\{\mathbf{x}_{N-k-1}\} = \mathbf{\Delta}^T_{N-k-1}.\{\mathbf{x}_{N-k}\}, \quad k = 0, N-1 \tag{49}$$

where

$\{\mathbf{x}_{N-k-1}\}$ is an aggregate of all of the states from which the states $\{\mathbf{x}_{N-k}\}$ are reachable in one step $k$. There is only one exception $\{\mathbf{x}_N\} = \mathbf{x}_N$, because the terminal state is only single.

Consequently, the backtracking system development is the following

$$\begin{aligned}
\{\mathbf{x}_{N-1}\} &= \mathbf{\Delta}^T_{N-1}.\mathbf{x}_N && (50)\\
\{\mathbf{x}_{N-2}\} &= \mathbf{\Delta}^T_{N-2}.\{\mathbf{x}_{N-1}\} = \mathbf{\Delta}^T_{N-2}\mathbf{\Delta}^T_{N-1}.\mathbf{x}_N && (51)\\
&\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\{\mathbf{x}_0\} &= \mathbf{\Delta}^T_0.\{\mathbf{x}_1\} = \mathbf{\Delta}^T_0\mathbf{\Delta}^T_1 \ldots \mathbf{\Delta}^T_{N-2}\mathbf{\Delta}^T_{N-1}.\mathbf{x}_N && (52)
\end{aligned}$$

However, using the constant matrix $\mathbf{\Delta}$ we have in general

$$\{\mathbf{x}_{N-k-1}\} = \mathbf{\Delta}^T.\{\mathbf{x}_{N-k}\}, \quad k = 0, N-1 \tag{53}$$

and the backtracking system development is the following

$$\begin{aligned}
\{\mathbf{x}_{N-1}\} &= \mathbf{\Delta}^T.\mathbf{x}_N && (54)\\
\{\mathbf{x}_{N-2}\} &= \mathbf{\Delta}^T.\{\mathbf{x}_{N-1}\} = \mathbf{\Delta}^T.(\mathbf{\Delta}^T.\mathbf{x}_N) = (\mathbf{\Delta}^T)^2.\mathbf{x}_N && (55)\\
&\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\{\mathbf{x}_0\} &= \mathbf{\Delta}^T.\{\mathbf{x}_1\} = \mathbf{\Delta}^T.((\mathbf{\Delta}^T)^{N-1}.\mathbf{x}_N) = (\mathbf{\Delta}^T)^N.\mathbf{x}_N && (56)
\end{aligned}$$

### 3.2.3 The control synthesis by means of intersection

The main problem of the control synthesis is that usually there are several possibilities how to proceed in any step of the system dynamics development. Consequently, there exists a tree of the possibilities of the system behaviour. The process of searching the most suitable path can

be tedious. The main idea of the approach presented here is to avoid the actual tree construction. Simultaneous utilising both the straight-lined approach and the backtracking one reduces the amount of computations. The procedure is the following:

- to proceed from the initial state $\mathbf{x}_0$ by the straight-lined approach
- to proceed from the desirable terminal state $\mathbf{x}_N$ by the backtracking approach
- to compare and intersect the aggregated states obtained by means of the straight-lined procedure and that obtained by means of the backtracking one. In such a way all possible trajectories from the given initial state to the desirable terminal one are found.

What is important is that the state reachability trees need not be generated in the form of actual graphs. It is sufficient to work with the numerical matrix $\mathbf{\Delta}$ of the OG-based model of DEDS and with three further constant matrices.

The intersection both the straight-lined tree and the backtracking one eliminates the complicated searching and yields all feasible trajectories from the initial state to the terminal one. Namely, the trajectories contain common fragments both of the trees. Because the relation between the trajectories and the control variables is known the control synthesis is finished by choosing the most suitable trajectory with respect to the control task specifications and by finding the corresponding sequence of the control vectors. The procedure of the control synthesis is the following:

1. to proceed from the initial state $\mathbf{x}_0$ by the straight-lined approach and enumerate the sequence $\{\mathbf{x}_0,\ {}^1\{\mathbf{x}_1\},\ \ldots,\ {}^1\{\mathbf{x}_N\}\}$ where the left upper index 1 denotes the straight-lined procedure. The elementary vectors are represented by the columns of the $(n\times(N+1))$-dimensional matrix $\mathbf{M}_1 = (\mathbf{x}_0, {}^1\{\mathbf{x}_1\}, \ldots, {}^1\{\mathbf{x}_N\})$
2. to proceed from the desirable terminal state $\mathbf{x}_N$ by the backtracking approach and enumerate the sequence $\{{}^2\{\mathbf{x}_0\},\ {}^2\{\mathbf{x}_1\},\ \ldots,\ \mathbf{x}_N\}$ where the left upper index 2 denotes the backtracking procedure. The elementary vectors are represented by the columns of the $(n \times (N+1))$-dimensional matrix $\mathbf{M}_2 = ({}^2\{\mathbf{x}_0\}, {}^2\{\mathbf{x}_1\}, \ldots, \mathbf{x}_N\})$
3. to make the column-to-column intersection $\mathbf{M} = \mathbf{M}_1 \cap \mathbf{M}_2$. The intersection of the corresponding columns is understood to be finding minima of their corresponding elements. The matrix $\mathbf{M} =$

$(\mathbf{x}_0, \{\mathbf{x}_1\}, \ldots, \{\mathbf{x}_{N-1}\}, \mathbf{x}_N)$, where $\{\mathbf{x}_i\} = min(^1\{\mathbf{x}_i\}, {}^2\{\mathbf{x}_i\})$, $i = 0, N$ with $^1\{\mathbf{x}_0\} = \mathbf{x}_0$, $^2\{\mathbf{x}_N\} = \mathbf{x}_N$, contains in its columns the vertices of the feasible trajectories (the aggregates of the feasible states)

4. to find the final set $\{T_i\}$ of transitions enabled in the step $i = 0, N-1$

$$\{T_i\} = {}^{bt}\{T_i\} \cap {}^{sl}\{T_i\} \tag{57}$$

$${}^{bt}\{T_i\} = \{{}^{bt}\{\mathbf{x}_i\}^+\};\ {}^{sl}\{T_i\} = \{{}^{sl}\{\mathbf{x}_i\}^+\} \tag{58}$$

$${}^{bt}\{\mathbf{x}_i\} = \mathbf{\Delta}_k^T.\{\mathbf{x}_{i+1}\} \tag{59}$$

$${}^{sl}\{\mathbf{x}_i\} = \mathbf{\Delta}_k.\{\mathbf{x}_i\} \tag{60}$$

where

$^{bt}\{\mathbf{x}_i\}^+$, $^{sl}\{\mathbf{x}_i\}^+$ denotes respectively the sets that consist only of the nonzero elements of the $^{bt}\{\mathbf{x}_i\}$ and $^{sl}\{\mathbf{x}_i\}$ ($^{sl}$-straight-lined; $^{bt}$-backtracking).

On this way the composed elements (like e.g. $a + b$ in general) express in (59) separating the elementary trajectories (i.e. forks) and in (60) their assembling (i.e. joins). Let as denote the former ones as $(a + b)^\vee$ and the latter ones as $(a + b)^\wedge$. The global set of t he enabled transitions is

$$\{T\} = \{\{T_0\}, \{T_1\}, ..., \{T_{N-1}\}\} \tag{61}$$

5. to choose the most suitable sequence $\{U\} = \{\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{N-1}\}$ of the control vectors by means of a suitable representation of knowledge about the control task specifications - e.g. by means of a knowledge base constructed by the methods presented in [4, 5, 6, 7].

**The applicability of the approach.** To utilise successfully the approach presented above it is necessary to inform about its universality in the sense of causality. In opposite case this aspect could stay hidden. On the base of the theorem proved in graph theory (see e.g. [27]) the solution of the control synthesis problem should be found after $n-1$ steps or less ($k \leq n-1$). However, sometimes it may be useful to observe a longer development of the system dynamics than it is necessary. Because of the causality principle any *shorter* feasible solution can be found very

simply because it is involved in the *longer* one. Namely, when during the intersection of the matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ the latter one is shifted to the left for one column we obtain another matrix ${}^{-1}\mathbf{M}$ with dimensionality $(n \times N)$. It is the following

$$ {}^{-1}\mathbf{M} = (\mathbf{x}_0, \{\mathbf{x}_1\}, \ldots, \{\mathbf{x}_{N-2}\}, \mathbf{x}_{N-1}) \tag{62} $$

where

$\mathbf{x}_{N-1} = \mathbf{x}_t$ (the same terminal state like before).

In general, the shifting (i.e. finding the $[n \times (N-k+1)]$-dimensional matrices ${}^{-k}\mathbf{M}, k = 1, 2, ...$) can continue until the intersections exist, i.e. until $\mathbf{x}_0 \in {}^{2}\{\mathbf{x}_0\}$ and simultaneously $\mathbf{x}_t \in {}^{1}\{\mathbf{x}_{N-k}\}$.

### 3.2.4 A general view on the approach

The adjacency matrix $\mathbf{\Delta}$ is the non-negative matrix defined e.g. in [20, 28, 30]. The necessary condition for the above procedure has to be fulfilled - namely, the terminal state $\mathbf{x}_N$ must be reachable from the initial state $\mathbf{x}_0$. It is not very difficult to test the reachability. For such a testing the result of the proved theorem [20, 26, 27] can be used. The test is based on computation of the $k$-th power (where $k$ is unknown before) of the OG adjacency matrix. Because we use the transpose $\mathbf{\Delta}$ of the original adjacency matrix, to obtain $\mathbf{\Delta}^k$ containing the first nonzero element $\delta_{ij}^k$ we have to multiply the matrix $\mathbf{\Delta}$ from the left. The nonzero element $\delta_{ij}^k$ of the matrix $\mathbf{\Delta}^k$ gives us information not only about the reachability of the $i$-th element of the state vector $\mathbf{x}_k$ from the $j$-th element of the state vector $\mathbf{x}_0$ but also about the number of the steps $k$ that have to be performed (more mathematical details can be found in literature [20, 26, 27]). How long is it necessary to compute the powers of the matrix? The question is answered in [27] - the exponent $k \leq n-1$. Of course, it is concerning only the reachability of a single position from another single one. The simultaneous reachability of several positions (the vector of the positions like the state vector in our case) from another vector of positions requires more steps. Because the higher powers of the adjacency matrix (that is nonnegative matrix) can be positive matrices, the following discussion is useful. The results on this way can create a base for a possible generalisation of the above introduced control synthesis procedure.

**A discussion about the exponent $k$ for the indecomposable adjacency matrix.** At the guess of the exponent $k$ in special cases the re-

sults published in [28, 30] concerning indecomposable non-negative $n \times n$-dimensional matrices can be utilised. The indecomposable matrix is defined in [20] as the matrix which is not decomposable. The matrix $\mathbf{A}$ is decomposable if it is of the following form (63) or if there exists a permutation matrix $\mathbf{P}$ such that $\mathbf{P}^T\mathbf{A}\mathbf{P}$ is of the form (63)

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{\emptyset} & \mathbf{A}_{22} \end{pmatrix} \tag{63}$$

where submatrices $\mathbf{A}_{ii}$, $i = 1, 2$ are square matrices. $\mathbf{\emptyset}$ is the zero matrix with the corresponding dimensionality. A real matrix $\mathbf{R}$ is non-negative if all its elements $r_{ij} \geq 0$. When a power of the indecomposable non-negative $n \times n$-dimensional matrix is a positive matrix (a real matrix $\mathbf{Q}$ is positive if all its elements $q_{ij} > 0$), the matrix is primitive (as it is mentioned in [28] this term was introduced by Frobenius in 1912). For the best upper bound of the exponent $k_{min}$ guaranteeing that corresponding power of the primitive matrix (for $n \geq 2$) is positive the inequality $k_{min} \leq \omega_n = (n-1)^2 + 1$ is valid - see [28, 30]. Hence, at least one common state can be found by both the straight-lined procedure and the backtracking one after such a number of steps. The proposed combined approach to the control synthesis seems to be powerful and may be promising for extension of its validity for a wider class of PN.

There are some new results for $n \geq 3$ in [21] concerning the minimal exponent. It is proved that if $k_{min} \geq \lfloor \omega_n/2 \rfloor + 2$ then the primitive *directed* graph has cycles of exactly two different lengths $i, j$ with $n \geq j > i$. Because in the graph theory - see e.g. [19] - the *oriented* and *directed* graphs are distinguished (the oriented graph is understood in [19] to be a directed graph having no symmetric pair of directed edges or in other words the directed graphs without loops or multiple edges) the new results are partially applicable for our case. The adjacency matrix of the OG-based model of DEDS can be decomposable in general (and consequently, it will be imprimitive). However, the considerations analogical to indecomposable matrix $\mathbf{A}$ can be done for the indecomposable submatrices $\mathbf{A}_{ii}$, $i = 1, 2$ of the decomposable matrix $\mathbf{A}$.

### 3.2.5 The illustrative example

Consider the same maze problem introduced above. Its structure is given on Fig. 22. The PN-based models for cat and mouse are given on Fig. 23 and the OG-based models on Fig. 24. The parameters of the PN-based models are introduced there too (the matrices ${}^c\mathbf{F}$, ${}^c\mathbf{G}$, ${}^m\mathbf{F}$, ${}^m\mathbf{G}$) as well

as the parameters of the OG-based models (${}^c\mathbf{\Delta}_k$ and ${}^m\mathbf{\Delta}_k$). Let us demonstrate on this example of DEDS the presented approach in details. Many particulars can be explained in such a way.

The control task specifications determine that the terminal state vector of the cat ${}^c\mathbf{x}_N$ is equal to the initial one

$$ {}^c\mathbf{x}_N = {}^c\mathbf{x}_0 = (0\,0\,1\,0\,0)^T \tag{64}$$

The terminal state vector of the mouse ${}^m\mathbf{x}_N$ is equal to the initial one as well

$$ {}^m\mathbf{x}_N = {}^m\mathbf{x}_0 = (0\,0\,0\,0\,1)^T \tag{65}$$

The powers of the adjacency matrix ${}^c\mathbf{\Delta}$ are the following

$$ {}^c\mathbf{\Delta} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad {}^c\mathbf{\Delta}^2 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} $$

$$ {}^c\mathbf{\Delta}^3 = \begin{pmatrix} 2 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & \boxed{\mathbf{1}} & 0 & 1 \\ 1 & 2 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad {}^c\mathbf{\Delta}^4 = \begin{pmatrix} 2 & 1 & 2 & 1 & 2 \\ 3 & 3 & 1 & 2 & 1 \\ 1 & 1 & \boxed{\mathbf{1}} & 2 & 1 \\ 3 & 2 & 1 & 3 & 1 \\ 1 & 2 & 1 & 1 & 1 \end{pmatrix} $$

$$ {}^c\mathbf{\Delta}^5 = \begin{pmatrix} 2 & 3 & 2 & 3 & 2 \\ 5 & 3 & 3 & 4 & 3 \\ 3 & 3 & \boxed{\mathbf{1}} & 2 & 1 \\ 5 & 4 & 3 & 3 & 3 \\ 3 & 2 & 1 & 3 & 1 \end{pmatrix} \quad {}^c\mathbf{\Delta}^6 = \begin{pmatrix} 6 & 5 & 2 & 5 & 2 \\ 7 & 7 & 5 & 6 & 5 \\ 5 & 3 & \boxed{\mathbf{3}} & 4 & 3 \\ 7 & 6 & 5 & 7 & 5 \\ 5 & 4 & 3 & 3 & 3 \end{pmatrix} $$

In the case of the mouse adjacency matrix ${}^m\mathbf{\Delta}$ they are as follows

$$ {}^m\mathbf{\Delta} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad {}^m\mathbf{\Delta}^2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} $$

$$
{}^{m}\mathbf{\Delta}^{3} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & \boxed{\mathbf{1}} \end{pmatrix}
{}^{m}\mathbf{\Delta}^{4} = \begin{pmatrix} 0 & 2 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 \end{pmatrix}
$$

$$
{}^{m}\mathbf{\Delta}^{5} = \begin{pmatrix} 0 & 0 & 2 & 0 & 2 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 \end{pmatrix}
{}^{m}\mathbf{\Delta}^{6} = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 2 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & \boxed{\mathbf{2}} \end{pmatrix}
$$

The transition matrices (i.e. the corresponding powers of the matrices $\mathbf{\Delta}$) yields information (see the bold elements ${}^{c}\delta_{3,3}$ of the corresponding powers of the matrix ${}^{c}\mathbf{\Delta}$) that the cat has single solutions with the length 3, 4, and 5 steps and three 6-step solutions. The mouse has (see the bold elements ${}^{m}\delta_{5,5}$ of the corresponding powers of the matrix ${}^{m}\mathbf{\Delta}$) the single solution of the length 3 and two 6-step solutions. This matrix is not primitive. It is a special matrix, where ${}^{m}\mathbf{\Delta}^{k+3} = 2.{}^{m}\mathbf{\Delta}^{k}$. Let us illustrate the proposed approach to the control synthesis. The straight-lined sequence of the aggregated states of the cat behaviour and that of mouse behaviour are stored, respectively, in the matrices ${}^{c}\mathbf{M}_1$ and ${}^{m}\mathbf{M}_1$.

$$
{}^{c}\mathbf{M}_1 = \begin{pmatrix} 0 & 1 & 0 & 0 & 2 & 2 & 2 \\ 0 & 0 & 1 & 1 & 1 & 3 & 5 \\ 1 & 0 & 0 & 1 & 1 & 1 & 3 \\ 0 & 0 & 1 & 1 & 1 & 3 & 5 \\ 0 & 0 & 0 & 1 & 1 & 1 & 3 \end{pmatrix}; {}^{m}\mathbf{M}_1 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 2 \end{pmatrix}
$$

Very analogically (using the powers of the transpose of the matrices ${}^{c}\mathbf{\Delta}$ and ${}^{c}\mathbf{\Delta}$) the backtracking sequence of the aggregated states of the cat and mouse are found

$$
{}^{c}\mathbf{M}_2 = \begin{pmatrix} 5 & 3 & 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 1 & 1 & 0 & 1 & 0 \\ 3 & 1 & 1 & 1 & 0 & 0 & 1 \\ 4 & 2 & 2 & 0 & 1 & 0 & 0 \\ 3 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}; {}^{m}\mathbf{M}_2 = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}
$$

Consider the situation when the the matrices ${}^{c}\mathbf{M}_1$ and ${}^{c}\mathbf{M}_2$ are overlapped as well as the matrices ${}^{m}\mathbf{M}_1$ and ${}^{m}\mathbf{M}_2$. Let us compare the

corresponding columns of the overlapped matrices and create their intersection. In such a way the resulting matrices ${}^{c}\mathbf{M}$ and ${}^{m}\mathbf{M}$ can be obtained.

$$ {}^{c}\mathbf{M} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}; {}^{m}\mathbf{M} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} $$

To assign the corresponding transitions to the elementary sections of trajectories of the cat the following has to be done

$${}^{bt}\{T_0\} = \{t_3^0, t_6^0\}^+; {}^{sl}\{T_0\}\{t_3^0\}^+$$

$${}^{bt}\{T_1\} = \{(t_1^1 + t_4^1)^\vee, t_7^1, t_8^1\}^+; {}^{sl}\{T_1\} = \{t_1^1, t_4^1\}^+$$

$${}^{bt}\{T_2\} = \{t_1^2, t_2^2, (t_5^2 + t_8^2)^\vee\}^+; {}^{sl}\{T_2\} = \{t_8^2, t_2^2, t_7^2, t_5^2\}^+$$

$${}^{bt}\{T_3\} = \{t_3^3, t_4^3, t_6^3, t_7^3\}^+; {}^{sl}\{T_3\} = \{(t_3^3 + t_6^3)^\wedge, t_2^3, t_7^3\}^+$$

$${}^{bt}\{T_4\} = \{t_1^4, t_8^4\}^+; {}^{sl}\{T_4\} = \{(t_1^4 + t_8^4)^\wedge, t_4^4, t_5^4\}^+$$

$${}^{bt}\{T_5\} = \{t_2^5\}^+; {}^{sl}\{T_5\} = \{t_2^5, t_7^5\}^+$$

$$\{T\} = \{[t_3^0], [(t_1^1 + t_4^1)^\vee], [t_2^2, (t_5^2 + t_8^2)^\vee], [(t_3^3 + t_6^3)^\wedge, t_7^3], [(t_1^4 + t_8^4)^\wedge], [t_2^5]\}$$

Very analogically the assignment for mouse trajectory sections can be done.

Let us illustrate the results on figures. The straight-lined and backtracking development of the cat is given on Fig. 27 and Fig. 28 respectively and those of the mouse on the Fig. 29. Although the cat graphs seem to be very complicated, from the numerical computation point of view they bring no special problems. Using the 6-step matrices the independent solutions for the cat and the mouse expressed on Fig. 30 are obtained. Comparing both the 6-step independent solution of the cat and that of the mouse and their confrontation with the control task specifications yield the final solution of the problem in question. Namely, $c_5^2$ is eliminated and $m_4^2$ is preferred because the room 5 is the place determined for the mouse comeback. Analogically, $m_1^2$ is eliminated and $c_2^2$ is preferred because the room 3 is the place determined for the cat

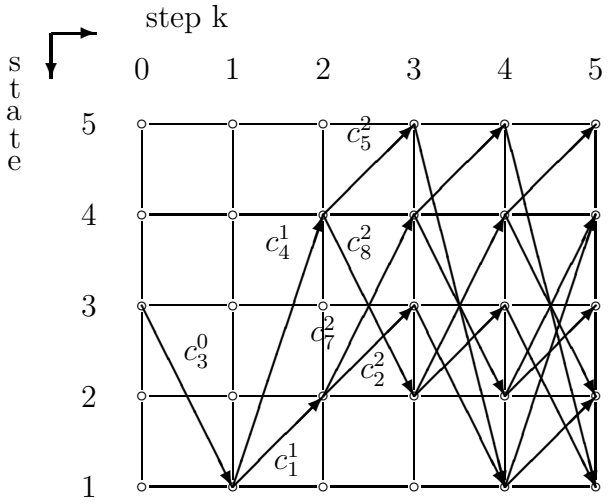


Figure 27: The graphical expression of the straight-lined development of the cat behaviour

comeback. The final solution of the cat and mouse control synthesis is given on Fig. 31.

Hence, the sequence of the final control vectors for the cat is the following

$$
\begin{aligned}
{}^c\mathbf{u}_0 &= (0,\,0,\,1,\,0,\,0,\,0,\,0,\,0)^T\,;\ {}^c\mathbf{u}_1 = (1,\,0,\,0,\,0,\,0,\,0,\,0,\,0)^T \\
{}^c\mathbf{u}_2 &= (0,\,1,\,0,\,0,\,0,\,0,\,0,\,0)^T\,;\ {}^c\mathbf{u}_3 = (0,\,0,\,1,\,0,\,0,\,0,\,0,\,0)^T \\
{}^c\mathbf{u}_4 &= (1,\,0,\,0,\,0,\,0,\,0,\,0,\,0)^T\,;\ {}^c\mathbf{u}_5 = (0,\,1,\,0,\,0,\,0,\,0,\,0,\,0)^T
\end{aligned}
$$

In order to satisfy the demand of the free movement also the following sequence is possible (however, it contains movement through the uncontrollable door and consequently, we can neither force the cat to pass through the door nor to prevent it from doing this).

$$
\begin{aligned}
{}^c\mathbf{u}_0 &= (0,\,0,\,1,\,0,\,0,\,0,\,0,\,0)^T\,;\ {}^c\mathbf{u}_1 = (0,\,0,\,0,\,1,\,0,\,0,\,0,\,0)^T \\
{}^c\mathbf{u}_2 &= (0,\,0,\,0,\,0,\,0,\,0,\,0,\boxed{1})^T\,;\ {}^c\mathbf{u}_3 = (0,\,0,\,1,\,0,\,0,\,0,\boxed{1},\,0)^T \\
{}^c\mathbf{u}_4 &= (1,\,0,\,0,\,0,\,0,\,0,\,0,\boxed{1})^T\,;\ {}^c\mathbf{u}_5 = (0,\,1,\,0,\,0,\,0,\,0,\,0,\,0)^T
\end{aligned}
$$

The sequence of the final control vectors for the mouse is the following

$$
\begin{aligned}
{}^m\mathbf{u}_0 &= (0,\,0,\,0,\,0,\,1,\,0)^T\,;\ {}^m\mathbf{u}_1 = (0,\,0,\,0,\,0,\,0,\,1)^T \\
{}^m\mathbf{u}_2 &= (0,\,0,\,0,\,1,\,0,\,0)^T\,;\ {}^m\mathbf{u}_3 = (0,\,0,\,0,\,0,\,1,\,0)^T \\
{}^m\mathbf{u}_4 &= (0,\,0,\,0,\,0,\,0,\,1)^T\,;\ {}^m\mathbf{u}_5 = (0,\,0,\,0,\,1,\,0,\,0)^T
\end{aligned}
$$

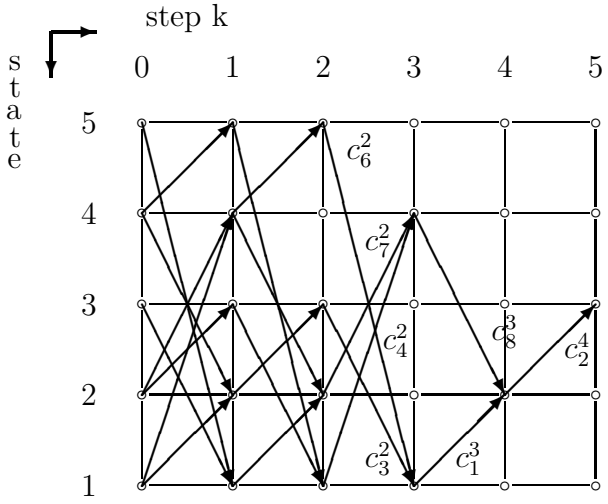


Figure 28: The graphical expression of the backtracking development of the cat behaviour

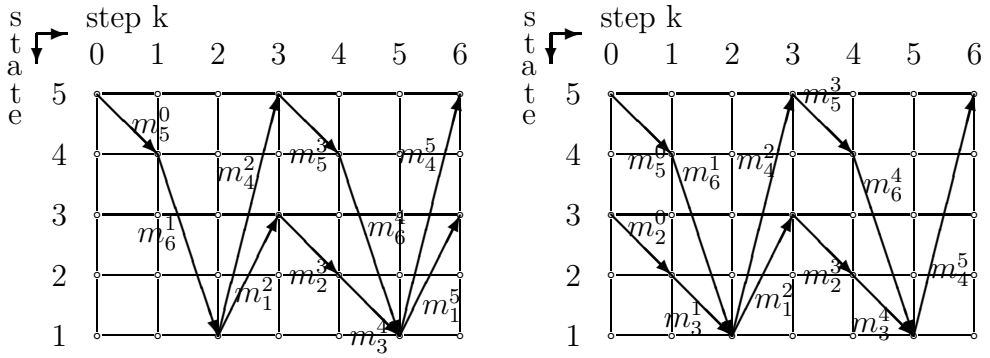


Figure 29: The graphical expression of the straight-lined (on the left) and the backtracking (on the right) development of the mouse behaviour

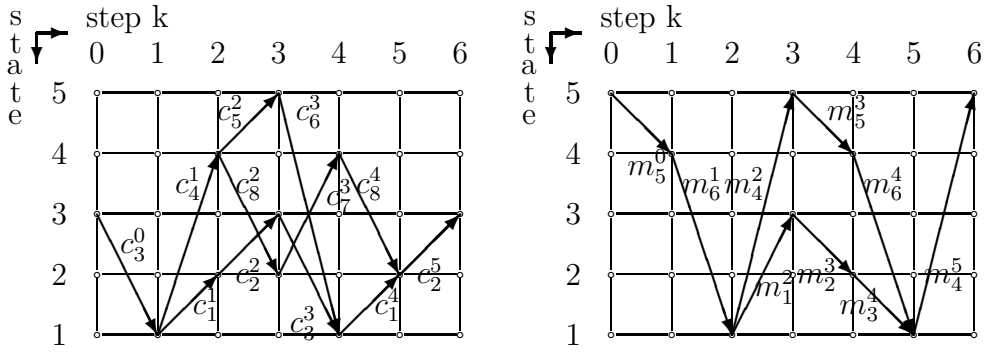


Figure 30: The independent 6-step solution of the cat control synthesis problem and that of mouse

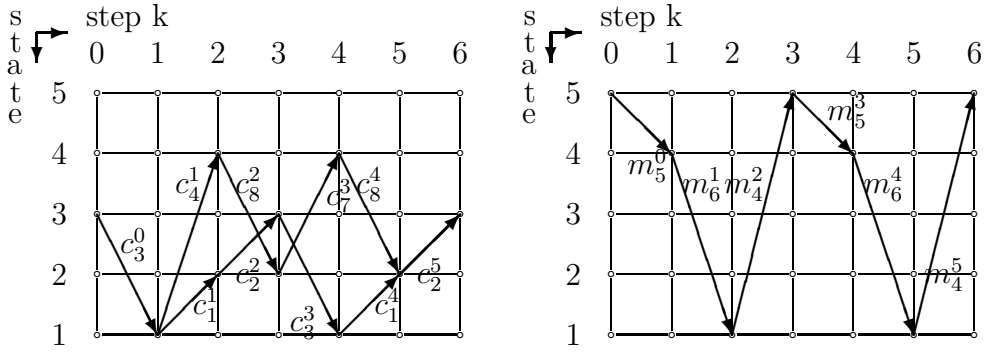


Figure 31: The final (mutually dependent) solution of the cat-and-mouse control synthesis problem

To explain better the principle of the proposed approach to the control synthesis, it is better to use the 5-step coincidence of the straight-lined and backtracking development of the cat behaviour. The resulting 5-step trajectory of the cat behaviour is single as follows

$$^{-1}[^{c}\mathbf{M}] = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

This matrix can be obtained by means of shifting the matrix $^{c}\mathbf{M}_2$ to the left for one column with respect to the matrix $^{c}\mathbf{M}_1$ before the intersection of the overlapped columns. After shifting the matrix $^{c}\mathbf{M}_2$ to the left once more the intersection of the overlapped columns yields the 4-step solution.

$$^{-2}[^{c}\mathbf{M}] = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

After further shifting to the left we have 3-step solution of the control synthesis problem.

$$^{-3}[^{c}\mathbf{M}] = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 32 expresses graphically the 5- and 4-step solutions of the cat, while the Fig. 33 expresses graphically the 3-step solution of both the cat and the mouse (obtained analogically). The 5- and 4-step solutions of the mouse do not exist.

## 3.3 Summary

By means of comparing both the 6-step straight-lined development of the system and the 6-step backtracking one we are able to find three 6-step independent solutions of the cat control synthesis problem. In

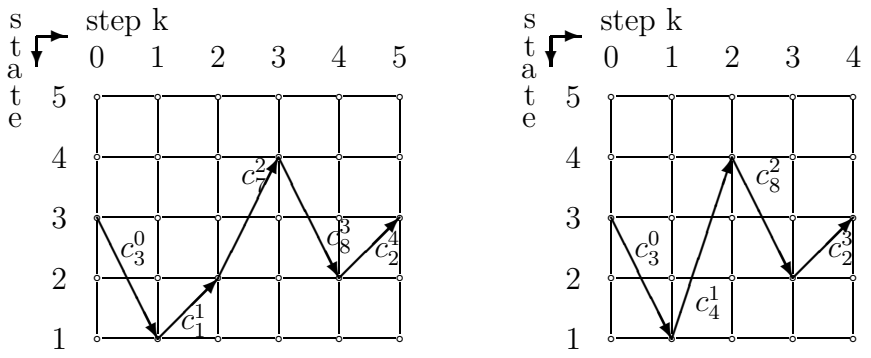


Figure 32: The 5-step solution (on the left) and 4-step solution (on the right)of the cat control synthesis problem

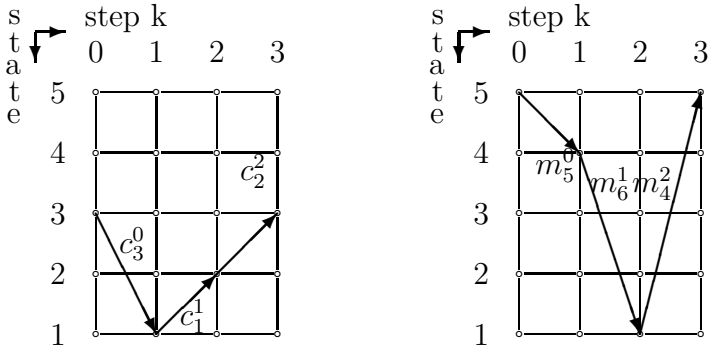


Figure 33: The 3-step solution of the cat (on the left) and mouse(on the right) control synthesis problem

addition to these in any step of the shifting we are able to find single independent solution of the cat. In such a way we have further three single independent solutions (5-, 4-, and 3-step ones). Analogically, two 6-step independent solutions and the single 3-step independent solution of the mouse control synthesis problem were found.

In order to denote the elementary sections of the trajectories by the corresponding transitions that have to be fired, the corresponding functional adjacency matrices ${}^{c}\mathbf{\Delta}_k$ and ${}^{m}\mathbf{\Delta}_k$ are utilised. Namely, they store information about the placement of the transitions.

To satisfy the prescribed control task specifications some of the independent solutions had to be eliminated and consequently, the final solutions (dependent on the control task specifications) were found. Con-

sequently, we have to compare the 6-step solution of the cat and that of the mouse and confront them with the control task specifications. The same can be done with the 3-step solutions. Hence, the 6-step solution of the cat-and-mouse problem given on Fig. 31 can be understood to be more complex like the 3-step solution given on Fig. 33. In addition to this we can see that the mouse solution is periodical with the period equal to 3 steps.

The main aim of the dissection of the cat-and-mouse problem solving into details was the endeavour to demonstrate the applicability of the proposed approach. Although the control task specifications were given only verbally, the approach makes possible to solve the control synthesis problem in analytical terms. However, if we want to solve the problem fully automatically knowledge about the control task specifications has to be represent in appropriate form. It is very important at the choice of the most suitable solution.

# 4 Conclusions

The problem of modelling and control of DEDS was solved in this report. The determining priority was the endeavour to find such models and control synthesis procedures that can be expressed in analytical terms. Two main kinds of the analytical models were presented - the PN-based models and the OG-based ones. While the PN-based models are suitable for the class of DEDS that can be described by OPN, the OG-based models are suitable only for state machines (i.e. special kind of PN). Both of the kinds of models were utilised at the analytical approaches to DEDS control synthesis.

Because the control task specifications are usually given verbally or in another non-analytical form, the suitable rule-based knowledge representation was used in order to quantify them. LPN and FPN were used on this way. Resulting KB was utilised in the proposed PN-based approach. The graphical tools for the DEDS modelling, for the KB creation, and even for the automatic knowledge-based control synthesis of DEDS were developed and described.

Several illustrative examples were solved in order to demonstrate the liveness of the proposed approaches.

# Acknowledgement

I should like to thanks Prof. Mogens Nielsen, director of BRICS, for creating very good conditions for my work. My thanks belong also to Dr. Uffe Engberg for his help at solving practical problems connected with finalisation of this report. Finally, I thank also to Mrs. Janne Christensen for the efficient administrative help.


# References


[1] Zoltán Bugár. *Logical Petri nets and fuzzy Petri nets as the means of knowledge representation (in Slovak).* Master Thesis (Supervisor: Čapkovič, F.). Department of Informatics, Faculty of Mathematics and Physics, Comenius University, Bratislava, Slovak Republic. 1998.

[2] František Čapkovič. *A Petri nets-based approach to the maze problem solving.* In *Discrete Event Systems: Modelling and Control.* (S. Balemi, P. Kozák and R. Smedinga, editors). Birkhäuser Verlag, Basel - Boston - Berlin. 1993. 173–179.

[3] František Čapkovič. *Knowledge-based control of DEDS.* In *Proc. of the 13th IFAC World Congress 1996, San Francisco, USA, June 30-July 5, 1996* (J.J. Gertler, J.B. Cruz and M. Peshkin, editors). Vol. J. Elsevier Science Ltd., Pergamon. paper J-3c-02.6. 1996. 347–352. also on Compact Disc.

[4] František Čapkovič. *Petri nets and oriented graphs in fuzzy knowledge representation for DEDS control purposes.* BUSEFAL. **69**, 1997. 21–30.

[5] František Čapkovič. *Fuzzy knowledge in DEDS control synthesis.* In *Proc. of the 7th International Fuzzy Systems Association World Congress - IFSA'97, Prague, Czech Republic, June 25-29, 1997.* (M. Mareš, R. Mesiar, V. Novák, J. Ramík, A. Stupňanová, editors). Academia, Prague, Czech Republic. 1997. 550–554.

[6] František Čapkovič. *An approach to knowledge-representation at control of DEDS.* In *Advances in Intelligent Systems* (F.C. Morabito, editor). IOS Press, Ohmsha, Amsterdam-Berlin-Oxford-Tokyo-Washington DC. 1997. 458–463.

[7] František Čapkovič. *Representation of fuzzy knowledge about control task specifications.* In *IT & KNOWS Information Technologies and Knowledge Systems. Proceedings of XV. IFIP World Computer Congress, August 31-September 4, 1998, Vienna, Austria, and Budapest Hungary.* (J. Cuena, editor). Riegelnik, Vienna. 1998. 215–228. also on Compact Disc.

[8] František Čapkovič. *Knowledge-based control synthesis of discrete event dynamic systems.* In *Advances in Manufacturing Systems. Decision, Control and Information Technology* ( S.G. Tzafestas, editor). Chapter 19. Springer Verlag, London. 1998. 195–206.

[9] František Čapkovič. *Knowledge-based control of production systems.* In *Proc. of the 6th European Congress on Intelligent Techniques & Computing - EUFIT'98, September 7-10, 1998, Aachen, Germany.* (H.J. Zimmermann, editor). ELITE Foundation, Aachen. Vol. 3. 1998. 1565-1569.

[10] František Čapkovič. *Fuzzy knowledge in control of manufacturing systems.* BUSEFAL. **75**, 1998. 4–17.

[11] František Čapkovič. *Automated solving of the DEDS control problems.* In *Multiple Approaches to Intelligent Systems* (I. Imam, Y. Kodratoff, A. El-Dessouki and M. Ali, editors). Lecture Notes in Artificial Intelligence. Springer, Berlin-Heidelberg-New York-Barcelona-Hong Kong-London-Milan-Paris-Singapore-Tokyo. Vol. 1611. 1999. 735–746.

[12] Peter Čapkovič. *Algorithm of the DEDS control synthesis and its program realization (in Slovak).* Master Thesis (Supervisor: Čapkovič, F.). Department of Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Bratislava, Slovak Republic. 1999.

[13] Peter Čapkovič. *Algorithm of the DEDS control synthesis and its program realization. User Handbook (in Slovak).* Department of Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Bratislava, Slovak Republic. 1999.

[14] D.Y. Chao, M. C. Zhou and D.T. Wang. *The knitting technique and Petri nets synthesis.* The Computer Journal. **37**, 1994. 67-76.

[15] S.M. Chen, J.S. Ke and J.F. Chang. *Knowledge representation using fuzzy Petri nets.* IEEE Transactions on Knowledge and Data Engineering. **2** No 3, 1990 311-319.

[16] Marián Csontos. *Timed and time Petri nets in modelling discrete event dynamic systems (in Slovak).* Master Thesis (Supervisor: Čapkovič, F.). Department of Informatics, Faculty of Mathematics and Physics, Comenius University, Bratislava, Slovak Republic. 2000.

[17] M. Diaz and P. Sènac. *The stream Petri nets: A model for timed multimedia information.* In *Proceedings of the 15th International Conference on Application and Theory of Petri Nets 1994, Zaragoza, Spain* (R. Valette, editor). Lecture Notes in Computer Science. Vol. 815. Springer. 1994. 219-238.

[18] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva and F.B. Vernadat. *Practice of Petri nets in Manufacturing.* Chapman & Hall. 1993.

[19] R. Diestel. *Graph Theory* Springer-Verlag, New York. 1997.

[20] M. Fiedler. *Special Matrices and their Using in Numerical Mathematics (in Czech).* SNTL Publishing House, Prague, Czechoslovakia. 1981.

[21] S. Kirkland, D.D. Olesky, and P. van den Driessche. *Digraphs with large exponent.* The Electronics Journal of Linear Algebra. **7**. 2000. 30–40.

[22] C.G. Looney, A.A., Alfize. *Logical controls via boolean rule matrix transformations.* IEEE Trans. on Syst. Man and Cybern. SMC-17. No 6. 1987. 1077–1081.

[23] C.G. Looney. *Fuzzy Petri nets for rule-based decision-making.* IEEE Trans. Syst. Man Cybern. SMC-18. No 1. 1988. 178–183.

[24] Csaba Nemes. *Graphical editor for the Petri nets creation (in Slovak).* Master Thesis (Supervisor: Čapkovič, F.). Department of Informatics, Faculty of Mathematics and Physics, Comenius University, Bratislava, Slovak Republic. 1997.

[25] J.L. Peterson. *Petri Net Theory and Modeling the Systems.* Prentice Hall, New York. 1981.

[26] J. Plesník. *Graph Algorithms (in Slovak).* VEDA. Bratislava, Czechoslovakia. 1983.

[27] F.P. Preparata, R.T. Yeh. *Introduction to Discrete Structures.* Addison-Wesley Publ. Comp., Reading, USA. 1974.

[28] J. Sedláček. *Introduction to Graph Theory (in Czech).* Academia, Prague, Czechoslovakia. 1977.

[29] S.G. Tzafestas, F. Čapkovič. *Petri net-based approach to synthesis of intelligent control for DEDS* In *Computer Assisted Management and Control of Manufacturing Systems* (S.G. Tzafestas, editor). Chapter 12. Springer Verlag, Berlin-Heidelberg-New York. 1996. 325–351.

[30] H. Wielandt. *Indecomposable non-negative matrices (in German).* Math. Zeitschrift. **52**. 1950. 642–648.

[31] W.M. Wonham, P.J. Ramadge. *On the supremal controllable sublanguage of a given language.* SIAM J. Control and Optimization. **25**. 1987. 637–659.

# Recent BRICS Report Series Publications

**RS-00-26** **František Čapkovič.** ***Modelling and Control of Discrete Event Dynamic Systems*****. October 2000. 58 pp.**

**RS-00-25** **Zoltán Ésik.** ***Continuous Additive Algebras and Injective Simulations of Synchronization Trees*****. September 2000. 41 pp.**

**RS-00-24** **Claus Brabrand and Michael I. Schwartzbach.** ***Growing Languages with Metamorphic Syntax Macros*****. September 2000.**

**RS-00-23** **Luca Aceto, Anna Ingólfsdóttir, Mikkel Lykke Pedersen, and Jan Poulsen.** ***Characteristic Formulae for Timed Automata*****. September 2000. 23 pp.**

**RS-00-22** **Thomas S. Hune and Anders B. Sandholm.** ***Using Automata in Control Synthesis — A Case Study*****. September 2000. 20 pp. Appears in Maibaum, editor,** ***Fundamental Approaches to Software Engineering: First International Conference,*** **FASE '00 Proceedings, LNCS 1783, 2000, pages 349–362.**

**RS-00-21** **M. Oliver Möller and Rajeev Alur.** ***Heuristics for Hierarchical Partitioning with Application to Model Checking*****. August 2000. 30 pp.**

**RS-00-20** **Luca Aceto, Willem Jan Fokkink, and Anna Ingólfsdóttir.** ***2-Nested Simulation is not Finitely Equationally Axiomatizable*****. August 2000. 13 pp.**

**RS-00-19** **Vinodchandran N. Variyam.** ***A Note on*** $\mathrm{NP} \cap \mathrm{coNP/poly}$**. August 2000. 7 pp.**

**RS-00-18** **Federico Crazzolara and Glynn Winskel.** ***Language, Semantics, and Methods for Cryptographic Protocols*****. August 2000. ii+42 pp.**

**RS-00-17** **Thomas S. Hune.** ***Modeling a Language for Embedded Systems in Timed Automata*****. August 2000. 26 pp. Earlier version entitled** ***Modelling a Real-Time Language*** **appeared in Gnesi and Latella, editors,** ***Fourth International ERCIM Workshop on Formal Methods for Industrial Critical Systems,*** **FMICS '99 Proceedings of the FLoC Workshop, 1999, pages 259–282.**